UNIVERSIDAD
DE LOS ANDES
MERIDA VENEZUELA

Big Data: Fundamentos, Estrategias, Gestión de datos (Hadoop, Map/Reduce, NoSQL), Análisis de datos (Spark, Splunk)

Jose Aguilar
CEMISID, Escuela de Sistemas
Facultad de Ingeniería
Universidad de Los Andes
Mérida, Venezuela

Big Data

Muchos datos se están recogiendo y almacenando

- Datos de la Web, comercio electrónico
- compras en almacenes/tiendas
- Bancos
- Redes Sociales

Big Data

- Google procesaba 20 PB al día (2008)
- Facebook tenía 2.5 PB de datos de usuario y procesaba 50 TB/día (2009)
- eBay tenía 6.5 PB de datos de usuario (5/2009)
  Large Hydron Collider del CERN (LHC) generaba 15 PB al año

Explosión de Datos

Air Bus A380

-1 billon de código
- cada motor genera 10 TB c/30 min

640TB por vuelo

Twitter generaba aproxim. 12 TB de datos/día

New York Stock intercambiaba 1TB de datos/día

Capacidad de almacenamiento se ha duplicado aproximadamente cada tres años desde la década de 1980

Explosión de Datos

Ciencia

- Bases de datos de astronomía, genoma, datos medioambientales, datos de transporte, ...

Humanidades y Ciencias Sociales

- Libros escaneados, documentos históricos, datos de las interacciones sociales, las nuevas tecnologías como el GPS ...

Comercio y Negocios

- Las ventas corporativas, transacciones del mercado de valores, el tráfico aéreo, ...

Entretenimiento

- Imágenes de Internet, películas de Hollywood, archivos MP3, ...

Medicina

- Resonancia magnética y tomografías computarizadas, registros de pacientes, ...

Big Data

“Volumen masivo de datos, tanto estructurados como no-estructurados, los cuales son **demasiado grandes y difíciles de procesa**r con las bases de datos y el software tradicionales" (ONU, 2012)

Big Data

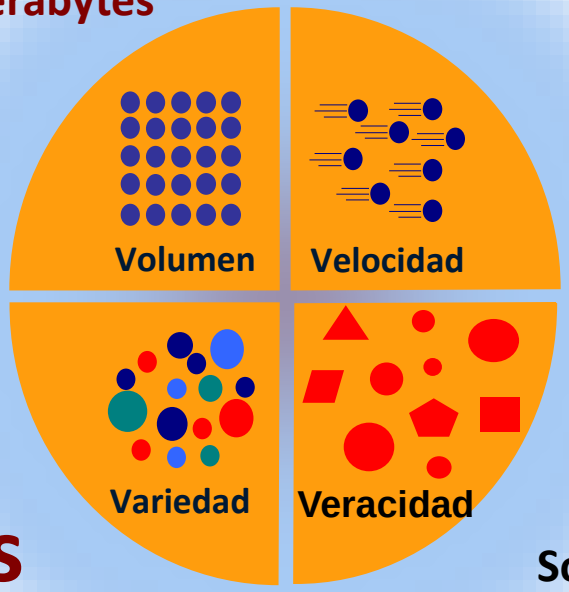
Los **grandes datos** permiten una mayor **inteligencia de negocios** mediante el **almacenamiento, el procesamiento y el análisis de datos** que se **ha ignorado** con anterioridad debido a las **limitaciones de las tecnologías tradicionales de gestión de datos**

Source: *Harness the Power of Big Data: The IBM Big Data Platform*

Big Data: Nueva Era de la Analítica

12+ terabytes
de Tweets
por día

5+ million
eventos comerciales por
segundo.

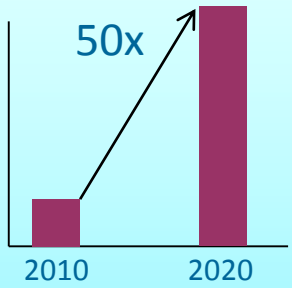


100's
de diferente
tipos de datos.

Solo **1 de 3**
tomadores de decisiones
confían en su información.

4 Características de Big Data

Eficiente procesamiento cada vez mayor de grandes **Volumenes**



35 ZB

En respuesta a la creciente **Velocidad**

30 Billones sensores RFID , etc.

Analizar la amplia **Variedad**

80% e los datos del mundo es no estructurado

Establecer la **Veracidad** de las fuentes de datos grandes

1 de 3 líderes de negocios no confían en la información que utilizan para tomar decisiones

Las cuatro dimensiones de Big Data

Volumen:

- Hoy en día, se estima que estamos produciendo 2,5 trillones de bytes de datos por día.

Para darse cuenta de lo mucho que esto es, pensar que un trillón es un millón de millones de personas.

- La velocidad a la que se produce de datos está en constante aumento.

- Se estima que más del 90% de los datos que hemos generados y recogidos hasta el momento se ha producido en los últimos 2 años.

- Aproximadamente el 90% es no estructurado, es decir, procedentes de fuentes que no son las bases de datos convencionales.

- Se producen a un ritmo que nunca han visto antes.

Por ejemplo, en 60 segundos hay más de 80 mil publicaciones en el muro en Facebook, 370 miles de llamadas de Skype, alrededor de 100 miles de tweets y más de 170 millones de mensajes de correo electrónico enviados.

Las cuatro dimensiones de Big Data

Volumen:

- Los datos de interés que se producen no sólo son de naturaleza electrónica o social de la red, pueden ser de todos los aspectos de la actividad humana:

Teléfonos que se compran, las transferencias de dinero, el paso de las personas en un punto de la ciudad, las fluctuaciones de la temperatura en una ciudad.

- Hemos dotado a nuestras vidas con una serie de dispositivos inteligentes que pueden supervisar todos los aspectos que hacemos

Por ejemplo, los teléfonos inteligentes recogen mensajes, llamadas telefónicas, y las imágenes que un usuario toma.

- Además, estos dispositivos inteligentes pueden poner en manos de un ciudadano, o un gerente todo un mundo de información,

Por ejemplo, darle al médico todos los incidentes en el mundo en el que se ha observado un conjunto similar de síntomas.

Las cuatro dimensiones de Big Data

Volumen:

- Los gobiernos y las empresas hoy en día se encuentran con grandes volúmenes de datos, que las tecnologías existentes no pueden manejar.
- Por lo tanto, hay una necesidad de tecnologías más avanzadas que pueden hacer frente a este gran volumen de datos.

Una definición de grandes volúmenes de datos

"grandes datos no se trata de tamaño, pero se trata de granularidad".

- La capacidad de los sistemas de software para identificar a los individuos y los datos personalizados es la implicación irónica de "grande".
- Es la capacidad de concentrarse en las minucias de la persona, en tiempo real.

Las cuatro dimensiones de Big Data

Velocidad:

- Los datos que recogemos hoy en día, en muchos casos llegan a los sistema de procesamiento a tasas muy altas.

- Este no es un problema, si la única tarea es almacenar los datos.

Pero si lo es si se quiere analizar y detectar una situación que se modela en los datos en tiempo de ejecución y tomar una decisión correctiva o cualquier otro tipo de acción inmediata.

- En el pasado, no se estaba al tanto de un problema antes de que alguien informara el problema.

En la era moderna, un problema se puede detectar en tiempo real.

Las cuatro dimensiones de Big Data

Velocidad:

Este tipo de datos normalmente se denominan corrientes/flujos (streams).

- Corrientes necesitan una gestión especial ya que uno
  - no tiene el lujo de almacenar los datos y revisarlos después.
  - no puede tomar sólo un aspecto de los datos a medida que llega y perder el resto.

- Cuando las tasas de los datos que llegan es relativamente lento, hay técnicas que se han desarrollado en las últimas décadas que permiten hacer inferencias y responder a las consultas sobre los datos de streaming.

- Hoy en día que las corrientes han llegado a ser tan grande en términos de velocidad y volumen, y el número de corrientes se multiplican también significativamente, el procesamiento de flujos de datos se está convirtiendo en un verdadero desafío.

Las cuatro dimensiones de Big Data

Variedad:

existe una necesidad no sólo de manejar grandes volúmenes de datos, sino también administrar una alta variedad de ellos,

Hábitos y estilo de vida

- **Los datos de las redes sociales:** publicaciones en el muro en Facebook, llamadas de Skype, tweets y mensajes de correo electrónico enviados.

- **Dispositivos en una ciudad:**
  - las cámaras en diferentes puntos de la ciudad permiten a los empleados de vigilancia ver un incidente en el momento en que sucede,
  - sensores inteligentes que son capaces de conocer la ubicación de los ciudadanos en una ciudad, el tráfico en una calle específica, la ubicación de cada autobús o tren,
  - la temperatura en diversos puntos de la ciudad,
  - medidores inteligentes miden el consumo eléctrico y reportan cada 15 minutos a la empresa de energía.

Smart city

- **Datos transaccionales:** Un número de transacciones de tarjetas de crédito debe desencadenar inmediatamente una alarma de actividades sospechosas y bloquear la tarjeta de espera de verificación.

Inteligenci de Negoci

Las cuatro dimensiones de Big Data

Veracidad:

La cantidad de confianza que uno pone en los datos.

- Dado que los datos proviene de fuentes que pueden no estar bajo nuestro control, pueden tener muchos problemas de calidad de datos, no pueden ser certificados, etc.

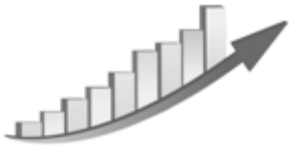
- Hay una necesidad de medir la confianza que ponemos en los datos, que a su vez significa la confianza que ponemos en las conclusiones y puntos de vista que hemos obtenido mediante el análisis de los datos.

Big Data

Los aspectos en que los usuarios quieren interactuar con sus datos …

- **Procesamiento:** Los usuarios tienen un mayor deseo de procesar y analizar todos los datos disponibles
- **Exploración:** Los usuarios quieren aplicar enfoques analíticos en el que el esquema se define en respuesta a la naturaleza de la consulta
- **Frecuencia:** Los usuarios tienen un deseo de aumentar la velocidad de análisis, con el fin de generar inteligencia de negocios más precisa y oportuna
- **Eficiencia:** necesidad de los usuarios para equilibrar la inversión en tecnologías y capacidades existentes, con la adopción de nuevas técnicas

El valor de Big Data



- **Predecir** el comportamiento del cliente en todos los ámbitos
- **Comprender** el comportamiento del cliente

- **Mejorar** la eficacia operativa
  - Máquinas / sensores: **predecir** fracasos, ataques a la red
  - Gestión de riesgos financieros: **reducir** el fraude, **aumentar** la seguridad

Reducir el costo de data warehouse

- Integrar las nuevas fuentes de datos sin aumentar el costo base de datos
- Proporcionar acceso en línea a "datos oscurOS’

Importancia de Big Data

Gobierno

- En 2012, el gobierno de Obama anunció financiamiento para la Investigación en Big Data

Sector privado

- Walmart maneja más de 1 millón de transacciones de los clientes cada hora, que se almacenan en sus bases de datos, que se estima contienen más de 2,5 petabytes de datos
- Falcon Credit Card Fraud Detection System protégé 2.100.000.000 cuentas en todo el mundo

Ciencia

- Gran Telescopio para Rastreos galáctico genera 140 terabytes de datos cada 5 días.
- Cálculos de Medicina como la decodificación del genoma

Los 5 clásicos Casos de uso en Big Data

Exploración
Encontrar, visualizar, comprender todos los grandes volúmenes de datos para mejorar la toma de decisiones

Tener una vista del cliente de 360o
Extender puntos de vista de los clientes existentes, mediante la incorporación de fuentes de información internas y externas adicionales

Extensión Inteligente de la Seguridad
minimar riesgo, detectar fraudes y supervisar la seguridad informática en tiempo real

Análisis de operaciones
Analizar una variedad de datos para mejorar resultados comerciales

Aumentar capacidades de Procesamiento de Datos
Integrar capacidades de big data y data warehouse para aumentar la eficiencia operativa

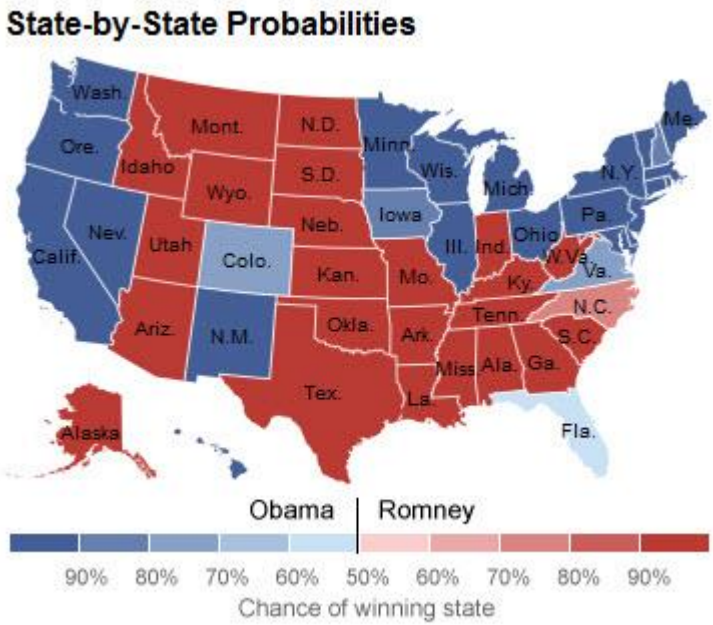
Ejemplo uso Big Data

Predicción de análisis de datos para Elección EE.UU. 2012

Drew Linzer, Junio 2012
332 para Obama,
206 para Romney

Nate Silver's,
Predijo que Obama tuvo la oportunidad de ganar un 86%
Predijo los 50 estados correctamente

Sam Wang, the Princeton Election Consortium
Determinó la probabilidad de la reelección de Obama



Ejemplos de uso de Big Data

IBM y el concurso de IBM Watson Deep Blue

- Un computador debería responder a las preguntas de conocimiento como las que los seres humanos responden normalmente.
- El secreto fue añadir un nivel para procesar enormes cantidades de información escrita en texto en lenguaje natural, principalmente la información de Wikipedia, y en poco tiempo a hacer las combinaciones necesarias para formular las respuestas.

Wal-Mart registra miles de transacciones de sus clientes

- Las analiza para entender la popularidad de los productos, y asociar con sus preferencias y hábitos.

Empresas financieras como American Express

- Están ejecutando extensos análisis y técnicas de inteligencia de negocio en las transacciones realizadas por sus clientes para tratar de entender sus intenciones.

Por ejemplo, para identificar potenciales clientes insatisfechos en los diferentes servicios financieros y predecir los clientes que desaparecerán en el futuro

Algunas experiencias internacionales a nivel gubernamental

Corea del Sur: “Plan Maestro de Big Data para la Implementación de una Nación Inteligente” (2013), del gobierno coreano.

Estados Unidos: “Iniciativa de I+D en Big Data” (2012), propuesta de la administración Obama, dirigido por la Oficina para la Ciencia y la Tecnología de la Casa Blanca.

Japón: Dentro de la primera estrategia de crecimiento del Japón del gobiernode Shinzo Abe (“Desatar el poder del sector privado hasta su máxima extensión”), se encuentra un plan básico para aprovechar Big Data” (Mayo 2012).

Comisión Estadística de Naciones Unidas: Seminario de Asuntos Emergentes en la 44°Sesión de la Comisión: Big Data para la Política, el Desarrollo y las Estadísticas Oficiales

Algunos Desafíos en Big Data

Integración Big Data es multidisciplinarios

- Menos del 10% de los grandes del mundo de datos son genuinamente relacionales
- Integración de datos en lo real es desordenado, esquema complejo
- Integración de Big Data y web semántica

El Triple Desafío

- Web de datos contiene 31 mil millones tripletas RDF, 446 million de ellas son enlaces RDF, 13 mil millones de datos del gobierno, 6 mil millones de datos geográficos, 4,6 mil millones de publicación, 3 mil millones de datos en ciencias biológicas

BTC 2011, Sindice 2011

Demostrar el valor de la Semántica: dejar que la tecnología DBMS maneje grandes volúmenes de datos heterogéneos, como los datos de enlace y RDF

Algunos Desafíos en Big Data

- Objetividad y precisión son **engañosas**
- Más Datos no siempre es sinónimo de **mejores datos**
- No todos los datos **son equivalentes**
- El hecho de que sea accesible no significa que **sea ético**
- El acceso limitado a los grandes datos crea **nuevas brechas digitales**
- ¿Qué sucede en un mundo de **transparencia radical**, con datos ampliamente disponibles?
- ¿Cómo **cambiaría su negocio** si utilizó los datos para grandes generalizada, en tiempo real?

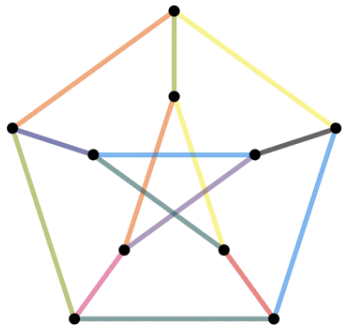
Cientificos de Datos

Un nuevo profesional con habilidades de gestión de datos, estadísticas, matemáticas, conocimiento en aprendizaje de máquina, para analizar los de datos.

- Los científicos tienen enormes cantidades de datos a analizar con el fin de convertirla en conocimiento útil para utilizarlos para tomar decisiones.
- Los científicos deben tener un buen conocimiento de algoritmos de análisis de datos que podrían ejecutar en los datos, con el fin de analizar los conjuntos de datos y entenderlos.

Suena como un informático que puede hacer estadísticas. ¿No ve lo científico aquí? .

Para más explicaciones sobre un científico de datos ver https://www.quantamagazine.org/20151214-graph-isomorphism-algorithm/.



Tecnología Big Data

Hadoop

- Bajo costo, arquitectura escalable fiable
- Éxito en computación distribuida

NoSQL

- Escala horizontal enorme y alta disponibilidad
- Altamente optimizado para recuperación y actualización
- Tipos
  - Documento
  - valor clave
  - bases de datos gráficas

RDBMS Análitico

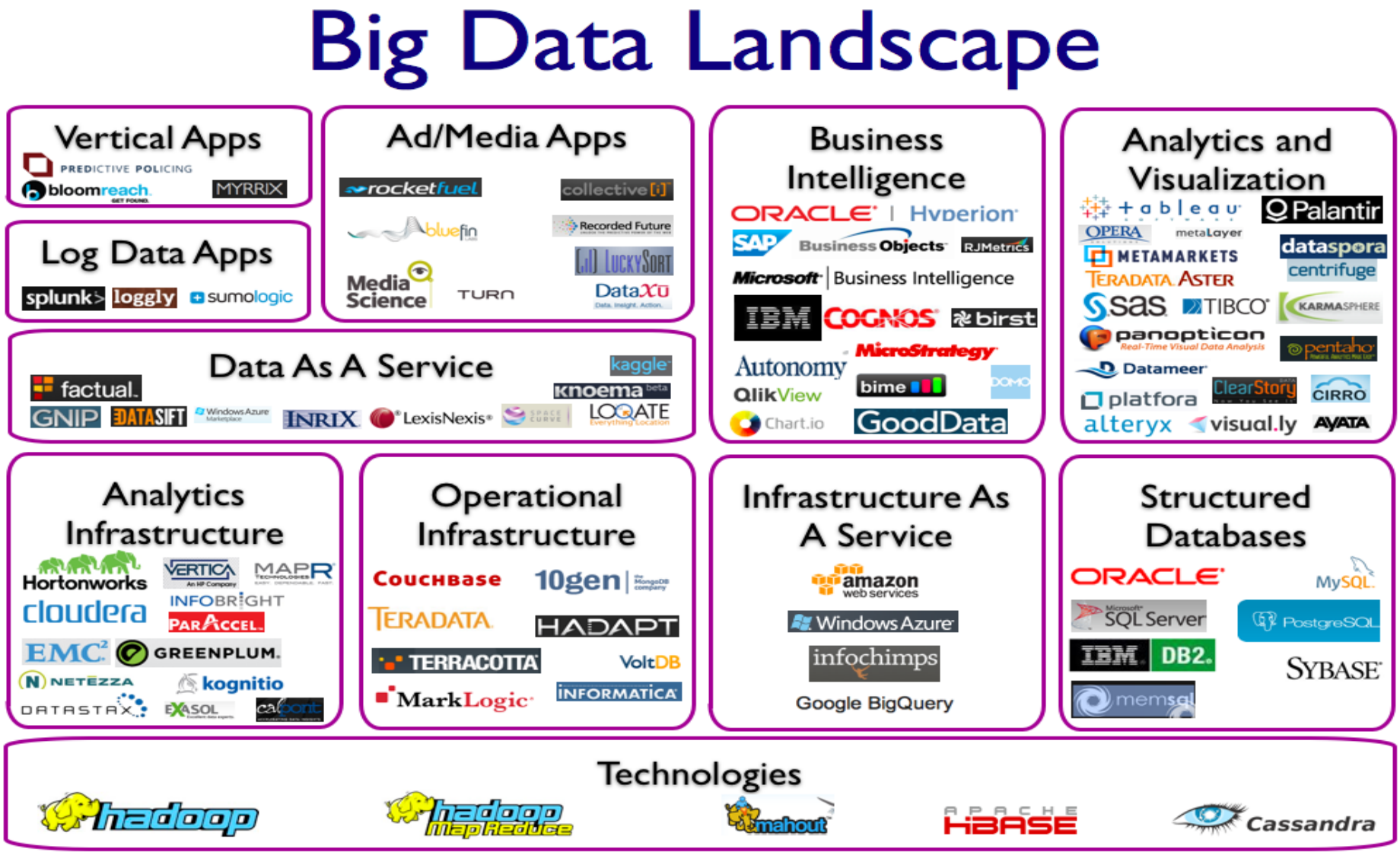
- Optimizado para cargas masivas de dato y cargas de trabajo de consulta y agregación intensas

Hadoop

NoSQL BDs

BD análiticas

PAISAJE DE BIG DATA



dave@vcdave.com

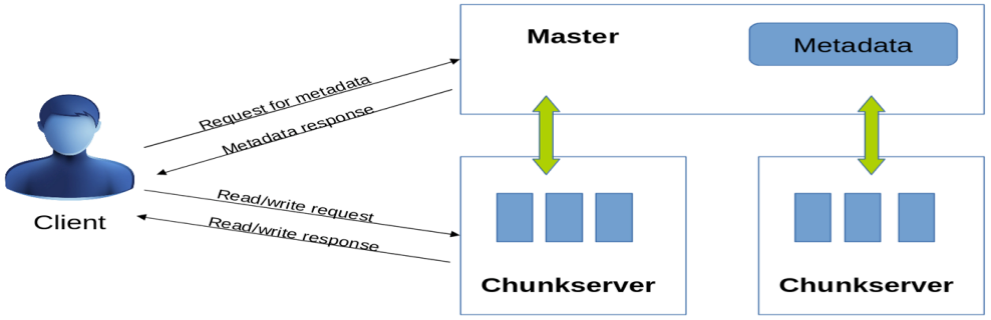
blogs.forbes.com/davefeinleib

Sistema de archivos distribuido

- Es una aplicación cliente/servidor en la que los clientes acceden y procesan los datos almacenados en el servidor.
- Uno o más servidores almacenan archivos que se pueden acceder, con los derechos adecuados de autorización, por cualquier cliente remoto en la red.
- Cuando el cliente solicita un archivo, el servidor envía una copia del archivo al usuario, que es capaz de trabajar con el archivo como si se tratara de su propio ordenador.
- Cuando el archivo es alterado por el cliente, las modificaciones se devuelven través de la red al servidor.

Google File System

- Múltiples nodos.
- Los nodos se dividen en dos tipos: un nodo maestro y varios Chunkservers.
- Cada archivo se divide en trozos de tamaño fijo y cada trozo se replica en varios sitios.
- Cliente accede a los trozos mediante la consulta primero el servidor maestro para ubicar los trozos deseados. Luego, contacta los chunkserve y recibe los datos directamente



Hadoop

Hadoop es un proyecto de Apache de código abierto que proporciona soluciones para la computación distribuida fiable y escalable.

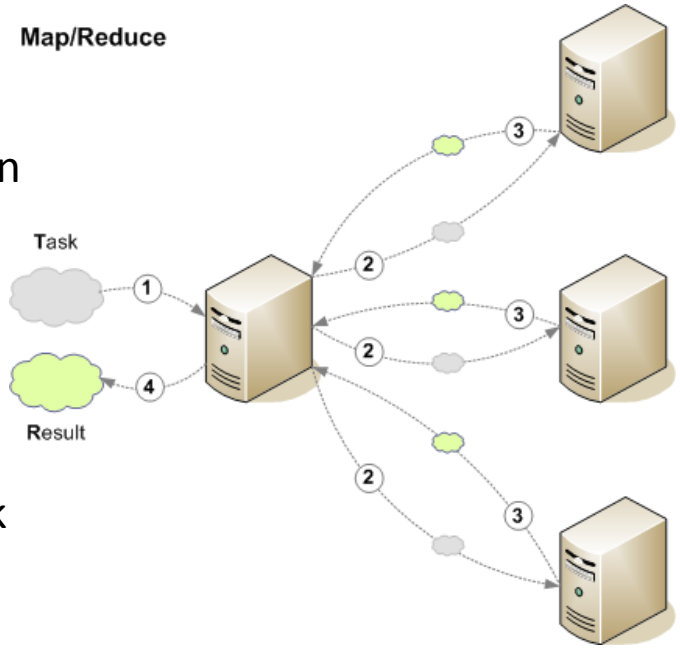
Hadoop es un sistema de archivos distribuido y motor de procesamiento de datos que está diseñado para manejar altos volúmenes de datos en cualquier estructura.

- Hadoop tiene dos componentes:
  - **El sistema de archivos distribuido Hadoop (HDFS),** que apoya a los datos en forma relacional estructurada, en forma no estructurada, y en cualquier otra forma
  - **El paradigma de programación MapReduce** para la gestión de aplicaciones en varios servidores distribuidos

- Se caracteriza por: **la redundancia, la arquitectura distribuida y el procesamiento paralelo**

Componentes Hadoop

- Hadoop Distributed File System (HDFS)
  - Almacenamiento redundante Masivo a través de un cluster básico
- MapReduce
  - Map: distribuye problema cálculo en el cluster
  - Reduce: Nodo maestro recoge las respuestas a todas las sub-problemas y las combina. Framework de programación para el desarrollo de aplicaciones y algoritmos.



- Varias distro disponible

Apache cloudera Hortonworks MAPR TECHNOLOGIES EMC GREENPLUM HD HADAPT amazon web services

Hadoop

HDFS (http://hadoop.apache.org/hdfs/): sistema de archivos distribuido, escalable y portátil.

- HDFS se compone de un NameNode y DataNode
- Un **NameNode** es un servidor maestro.
- **NameNode** gestiona el espacio de nombres del sistema de archivos y está vinculado a **DataNode** para gestionar el almacenamiento de datos.
- Los archivos se dividen en bloques de un tamaño fijo, que se distribuyen en uno o varios **DataNodes**.

Esta estructura implica que los bloques de un archivo se pueden guardar en varias máquinas.

- Los **DataNodes** realizan la creación, eliminación, y la replicación de bloques, según las instrucciones procedentes del **NameNode**

Hadoop

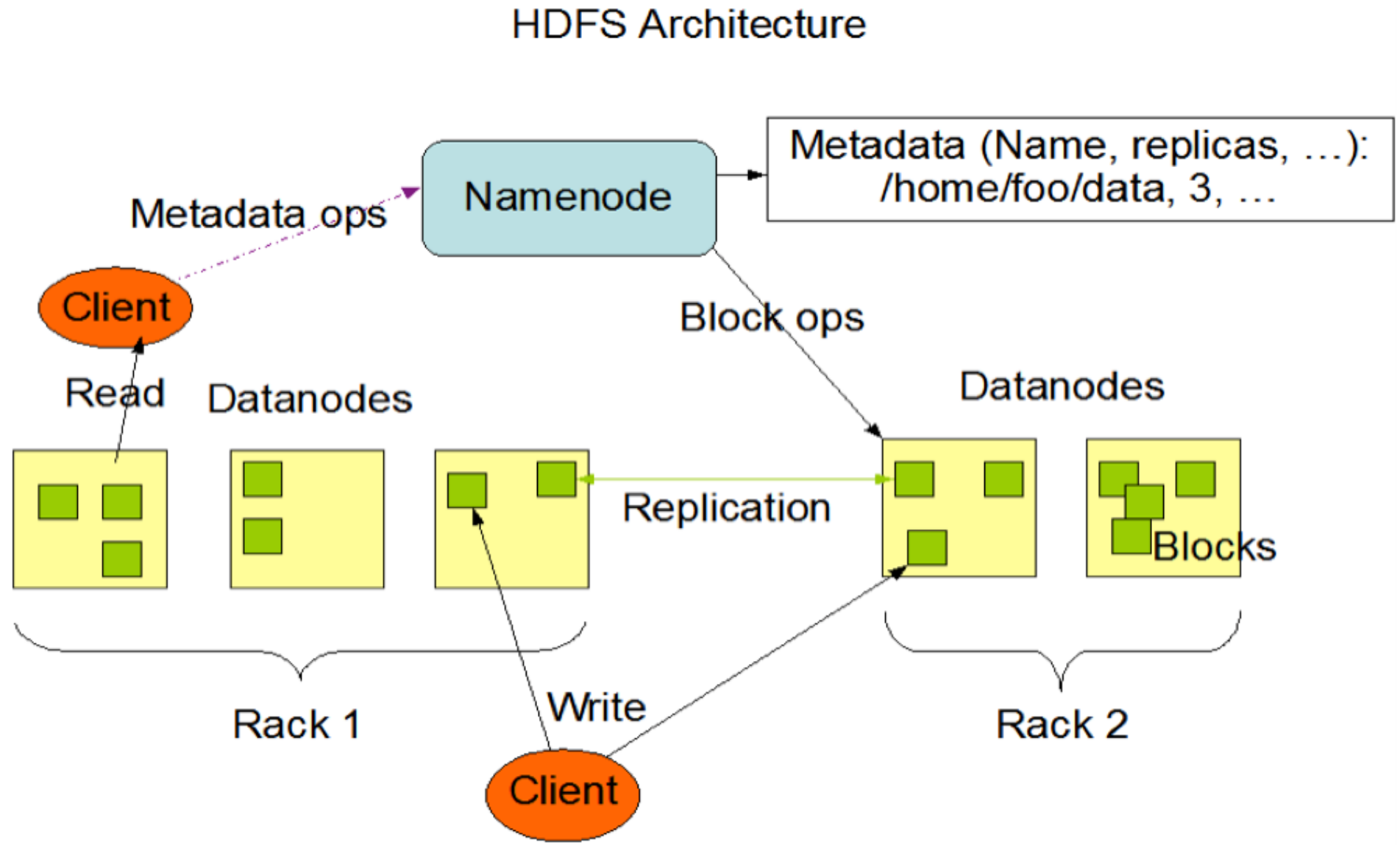
HDFS (http://hadoop.apache.org/hdfs/): sistema de archivos distribuido, escalable y portátil.

- Una desventaja importante es la pérdida de archivos debido a la falta de disponibilidad de una máquina.

HDFS resuelve este problema mediante la replicación de cada bloque de la mayoría de los nodos de la máquina.

- La información en los metadatos consiste en la partición de los archivos de en bloques y en la distribución de estos bloques en diferentes DataNodes.

Hadoop


HDFS Architecture
Namenode
Metadata (Name, replicas, …):
/home/foo/data, 3, …
Metadata ops
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Rack 1
Write
Client
Rack 2

NameNode y DataNodes

- El **NameNode y DataNode** son software diseñado para ejecutarse en máquinas que suelen ejecutar en Linux.

- **HDFS** se construye utilizando Java;
  Cualquier PC que soporte Java puede ejecutar NameNode o DataNode.

- Un despliegue típico es un **PC dedicado** que sólo ejecuta NameNode.

- Cada una de **los otros PC en el clúster** ejecutan una instancia de DataNode.

En la arquitectura pueden haber múltiples DataNodes en la misma máquina.

- La existencia de una sola NameNode en un clúster **simplifica la arquitectura** del sistema.

- El NameNode es el árbitro y repositorio para todos los **metadatos del HDFS**.

NameNode y DataNodes

HDFS tiene una arquitectura maestro/esclavo.

- Un clúster HDFS consta de una sola NameNode, servidor que **regula el acceso a los archivos de los clientes**.
- Hay una serie de DataNodes, por lo general uno por cada nodo del clúster, que **gestionan el almacenamiento conectado a los nodos que se ejecutan en él**.

HDFS expone un espacio de nombres en un sistema de archivos y permite que los datos de usuario se almacenen en archivos.

- Internamente, un archivo **se divide en uno o más bloques** y estos bloques **se almacenan** en un conjunto de DataNodes.

- **El NameNode**
  - Ejecuta operaciones en el sistema de archivos como **abrir, cerrar y cambiar el nombre de archivos y directorios**.
  - Determina la **asignación de bloques** en los DataNodes.

- **Los DataNodes**
  - Son responsables de **leer y escribir las peticiones de los clientes en el sistema de archivos**. Los DataNodes
  - Llevan a cabo la **creación, eliminación, y la replicación de bloques**, según instrucciones de la NameNode.

MRv1 y MRv2 Hadoop

- El nodo principal consiste en una **JobTracker, TaskTracker, NameNode y DataNode**.
- Cada nodo esclavo consiste en una **DataNode y TaskTracker**.

- Para los propósitos de tolerancia a fallos, **NameNodes secundarias se pueden usar para replicar los datos de la NameNode**.

- El JobTracker (Resource Manager) asume la responsabilidad de:
  - **la distribución de la configuración de software** a los nodos esclavos (resource management)
  - **la programación de tareas a los nodos TaskTracker** disponibles (job scheduling/monitoring)
  - **su seguimiento y la reasignación de tareas** a los nodos TaskTracker si es necesario (job scheduling/monitoring).
  - proporcionar la información de estado y de diagnóstico para el cliente.
- El TaskTracker (Node Manager) **ejecuta la tarea** como lo define el JobTracker.

VERSIONES HADOOP

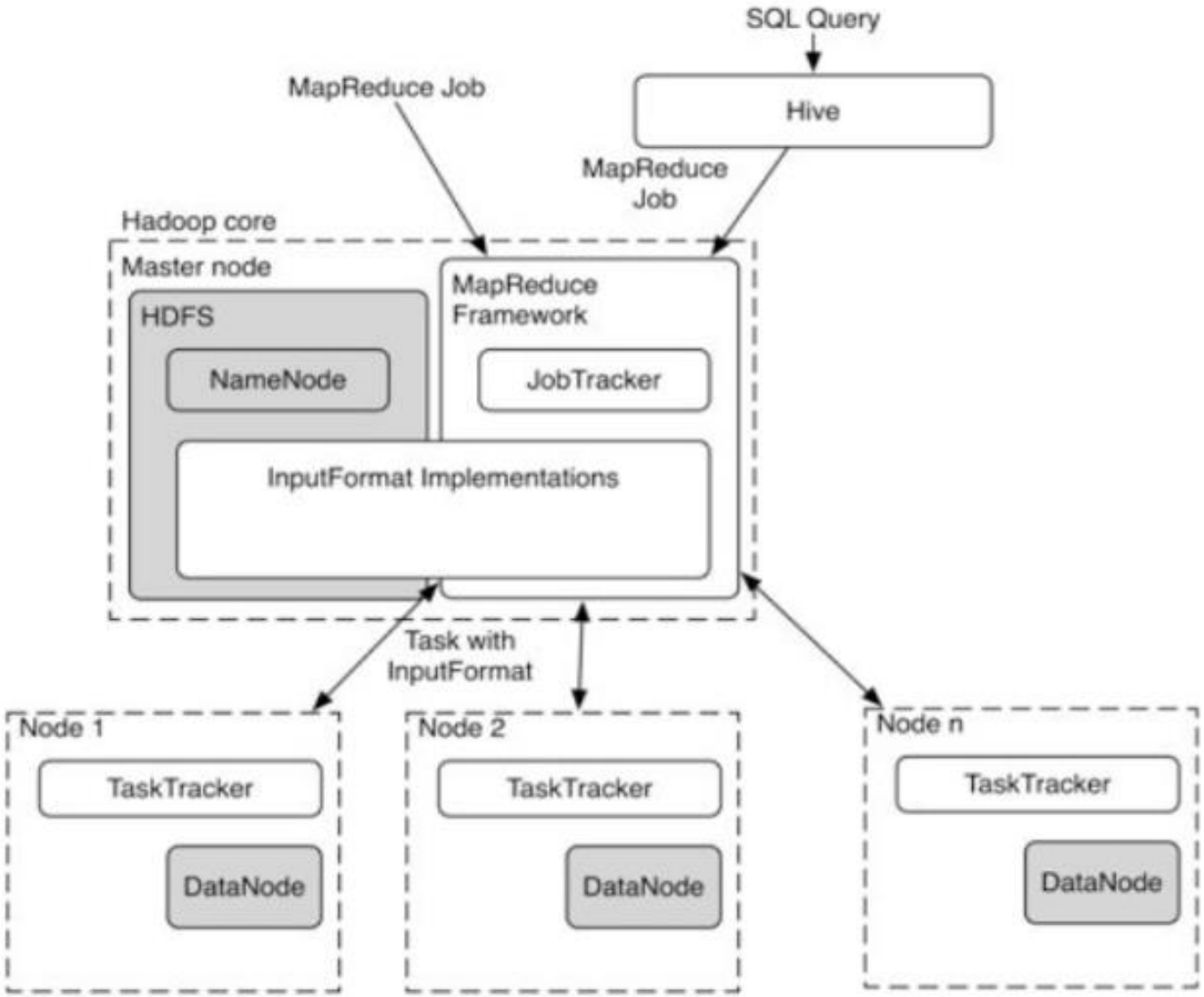
Hadoop 1.0: A pesar de existir ya la versión 2.0 de Hadoop, la primera versión aún es bastante utilizada por muchos desarrolladores al ser la más sólida y estable. El proyecto original de Hadoop se construyó sobre tres bloques fundamentales:

- **HDFS**
- **MapReduce**
- **Hadoop Common**: es un conjunto de las principales librerías y utilidades que la mayoría de proyectos Hadoop utilizan.

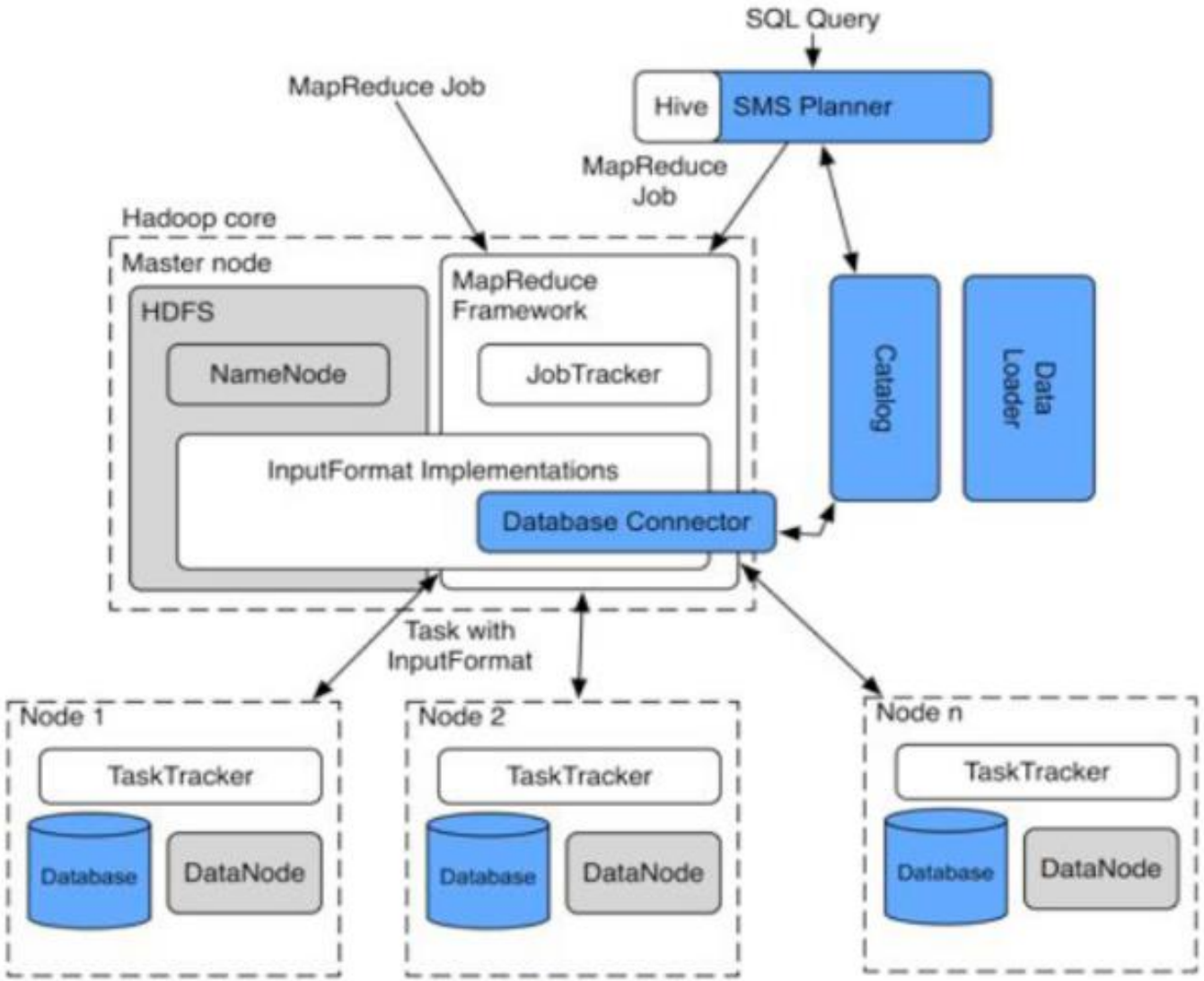
Hadoop 2.0: La segunda versión de Hadoop parte con la base de Hadoop 1.0 y añade -y modifica también- algunas características de sus módulos para tratar de resolver algunos de los problemas que tenía y mejorar el rendimiento del sistema. El proyecto Hadoop 2.0 está dividido esta vez en cuatro módulos:

- **Hadoop Common**
- **Hadoop Distributed File System (HDFS)**
- **Hadoop YARN**: un *framework* para la gestión de aplicaciones distribuidas y de recursos de sistemas distribuidos.
- **Hadoop MapReduce**: el sistema de procesamiento principal, que esta vez se ejecuta sobre YARN

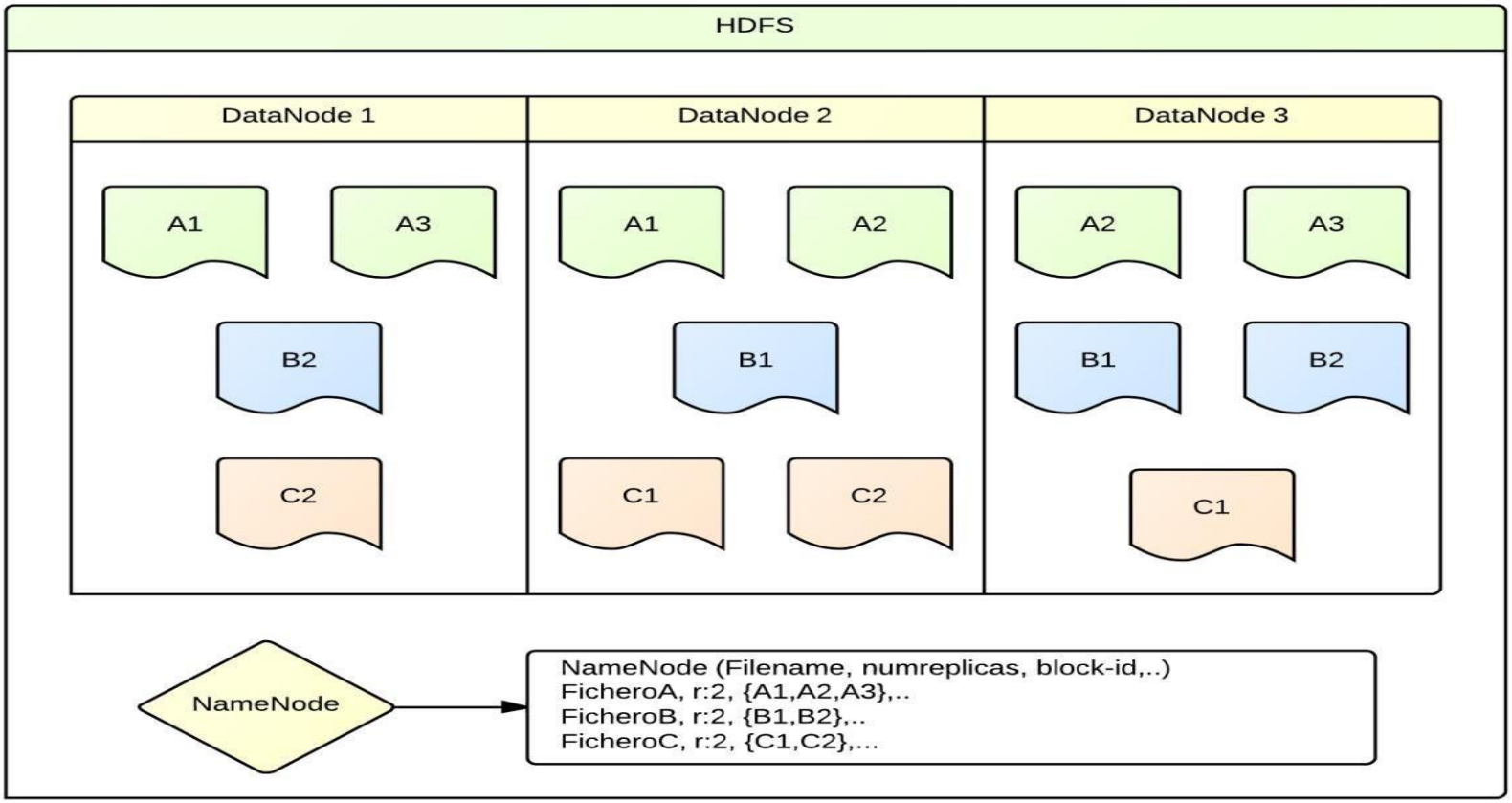
Arquitectura de Hadoop



Arquitectura de Hadoop



ARQUITECTURA HDFS



Ejemplo de replicación y división de los ficheros en bloques de tres ficheros (FicheroA, FicheroB y FicheroC) con factor de replicación dos. El NameNode guarda los metadatos del sistema de ficheros.

MapReduce

Marco para el tratamiento de problemas altamente distribuidos, con grandes volúmenes de datos, utilizando un gran número de computadores.

- Por ejemplo, Google para el procesamiento de grandes cantidades de datos en bruto, por ejemplo, los logs de la web.
- Los datos son tan grande, que debe ser distribuido a través de miles de máquinas con el fin de ser procesados en un período razonable de tiempo.
- Esta distribución implica computación paralela ya que los mismos cálculos se realizaron en cada CPU, pero con un conjunto de datos diferente.

MapReduce es una abstracción que permite realizar cálculos sencillos, al tiempo que oculta los detalles de paralelización, distribución de los datos, equilibrio de carga y tolerancia a fallos.

MapReduce

Funciones de base de MapReduce :

- **La función Map** toma como entrada un par y produce un conjunto de pares de claves y valores intermedios.
- **MapReduce** agrupa todos los valores intermedios asociadas a una misma clave intermedia y los pasa a la función Reducir.
- **La función Reduce** acepta la clave intermedio y un conjunto de valores para ella.
- **Los combina** para formar un posible conjunto de valores más pequeño.

MapReduce

Opera con pares <clave, valor>

- Ve la entrada como un conjunto de pares <clave, valor>
- Produce un conjunto de pares <clave, valor> como la salida, posiblemente de diferentes tipos.

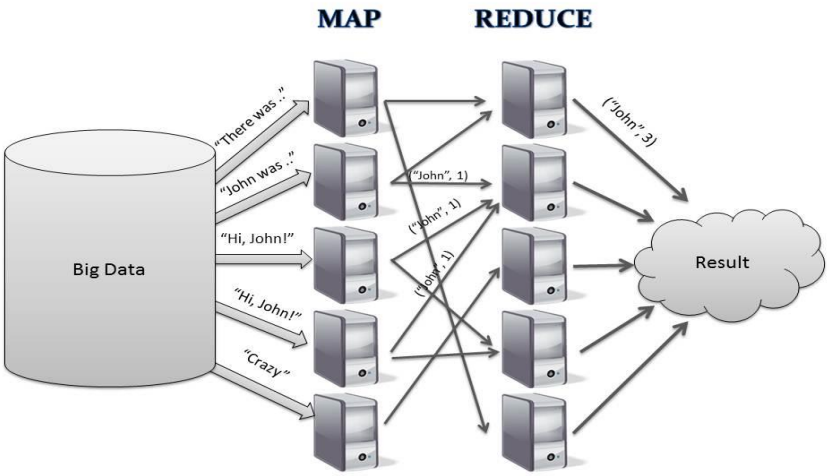
Tipos de Entrada y salida en un trabajo MapReduce:

(Entrada) <k1, v1> -> Map -> <K2, v2> -> reduce -> <K3, v3> (Salida)

	Input	Output
Map	<k1, v1>	list (<k2, v2>)
Reduce	<k2, list(v2)>	list (<k3, v3>)

FUNDAMENTOS DE MapReduce

¿Qué es MapReduce?



Es un framework que proporciona un sistema de procesamiento de datos paralelo y distribuido.

Utiliza el sistema de archivos distribuido HDFS.

Tiene una arquitectura maestro / esclavo.

Cuenta con un servidor maestro o **JobTracker** y varios servidores esclavos o **TaskTrackers**, uno por cada nodo del clúster.

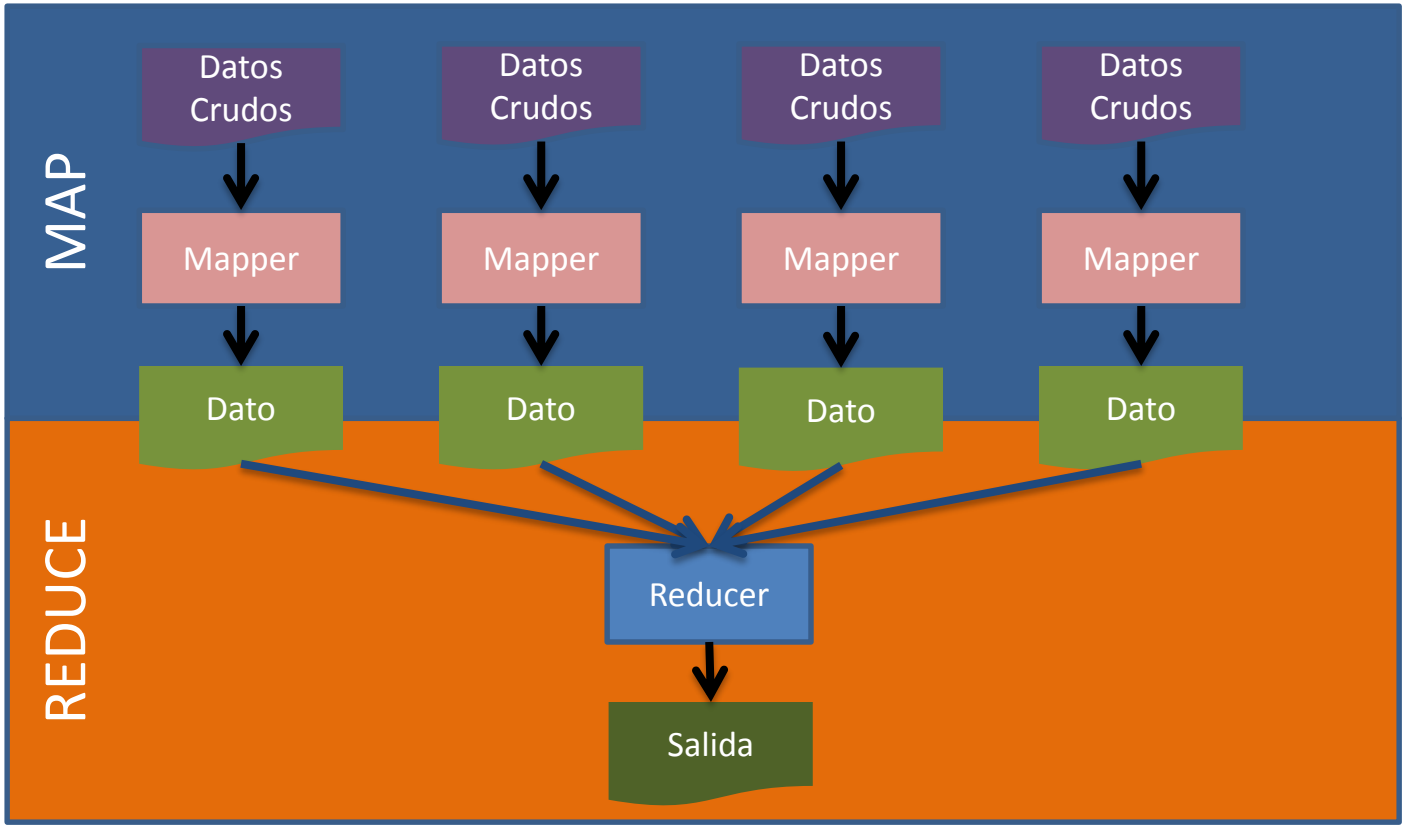
El **JobTracker** es el punto de interacción entre los usuarios y el framework MapReduce.

1. Los usuarios envían trabajos MapReduce al **JobTracker**, que los pone en una cola de trabajos pendientes y los ejecuta en el orden de llegada.

2. El **JobTracker** gestiona la asignación de tareas y delega las tareas a los TaskTrackers.

3. Los **TaskTrackers** ejecutan tareas bajo la orden del JobTracker y también manejan el movimiento de datos entre la fase Map y Reduce.

Map + Reduce = Extrae, Carga + Transforma



MapReduce

Paralelización automática:

- Dependiendo del tamaño de los datos de entrada sin procesar => instancia de múltiples tareas MAP

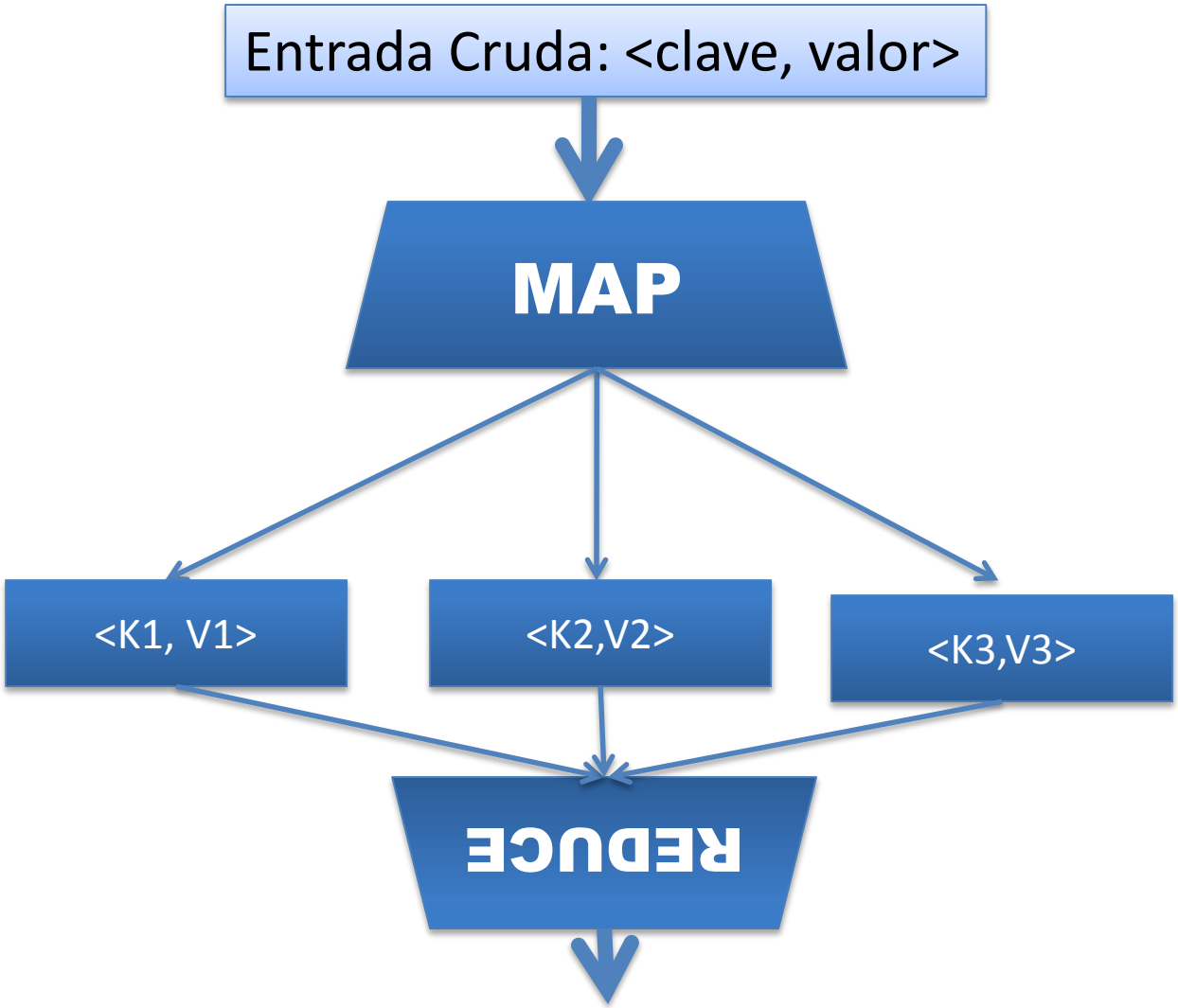
- Del mismo modo, dependiendo de la cantidad de producto intermedio <clave, valor> particiones => instancian múltiple tareas REDUCE

En tiempo de ejecución:

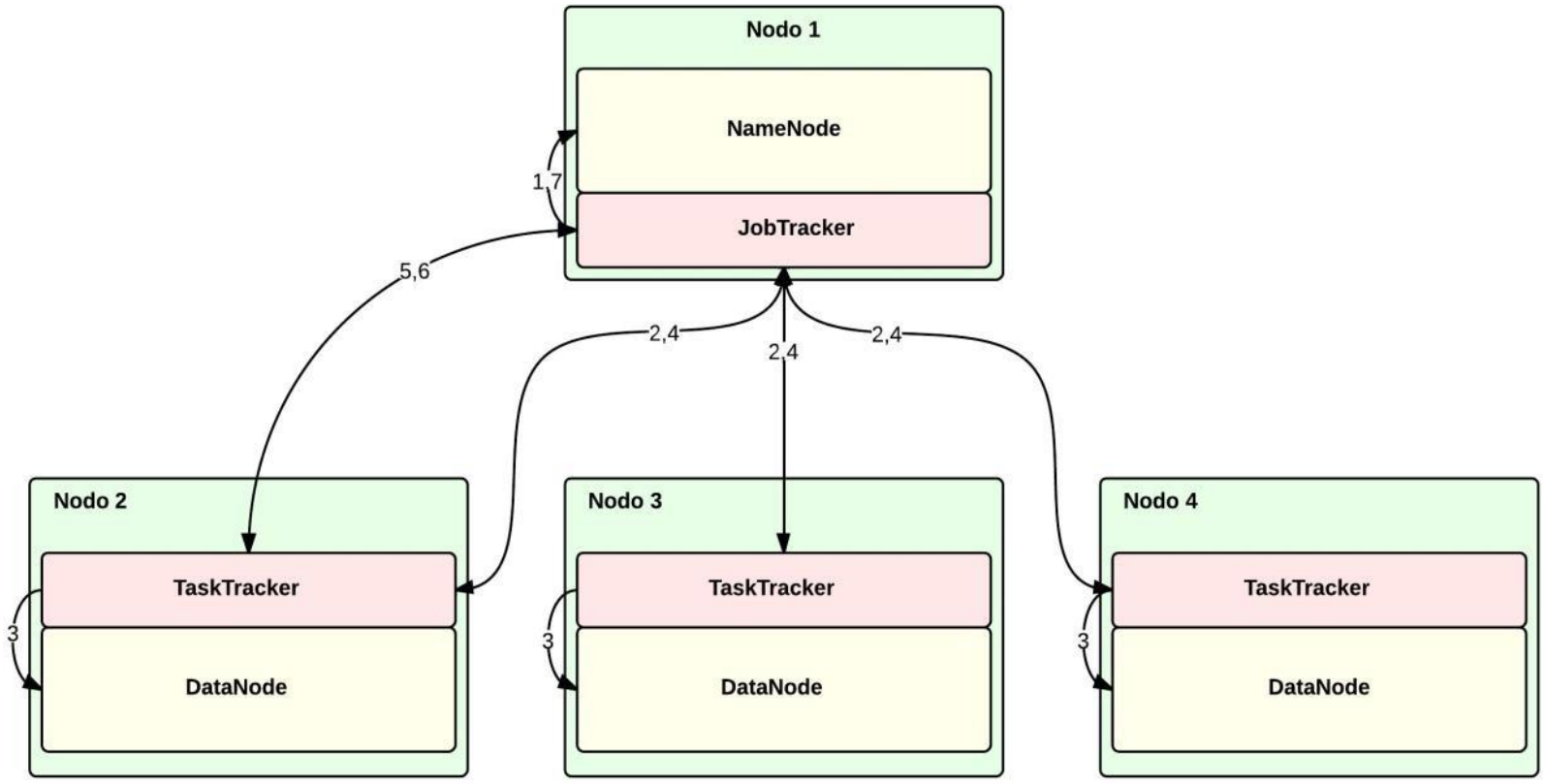
- Partición de datos
- Planificación de tareas
- Manejo de fallas
- Gestión de la comunicación entre máquinas

Completamente transparente para el programador/analista/usuario

MapReduce

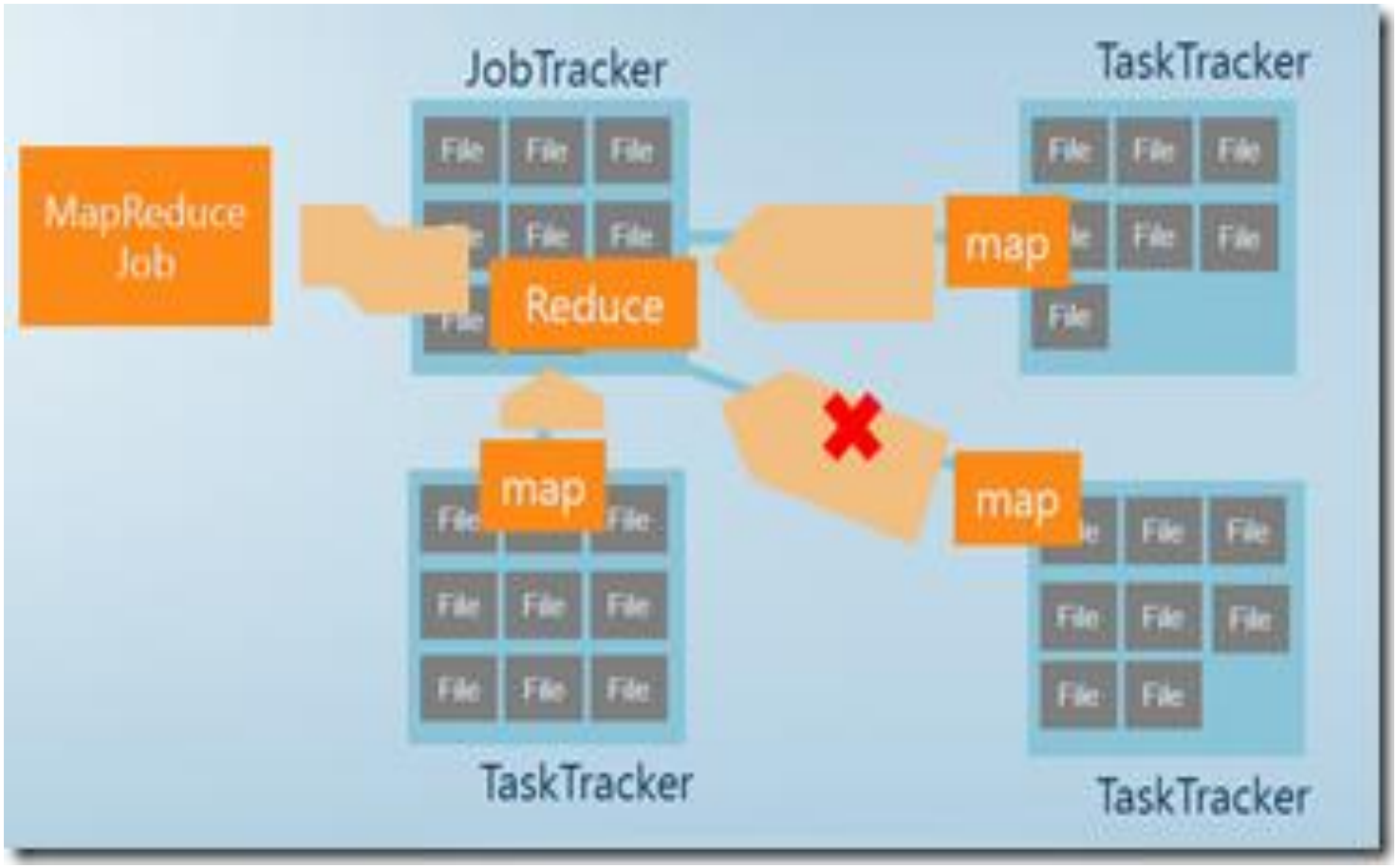


ARQUITECTURA MAPREDUCE

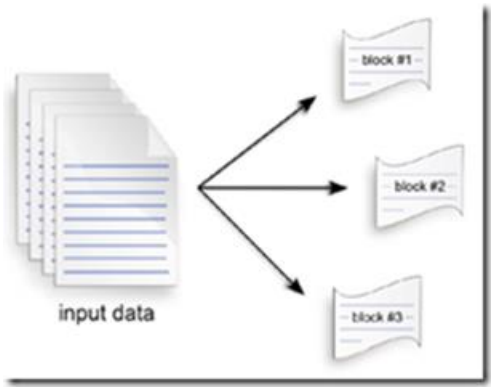


Fluo de trabajo para la ejecución de un trabajo MapReduce en una arquitectura Hadoop con cuatro nodos: tres de trabajo (DataNode+TaskTracker) y uno de gestión (NameNode+JobTracker).

ARQUITECTURA MAPREDUCE



Función Map( )



Operación Split

- La función Map recibe como parámetros un par (clave, valor) y devuelve una lista de pares.
- Esta función se encarga del mapeo y se aplica a cada elemento de la entrada de datos, por lo que se obtendrá una lista de pares por cada llamada a la función Map.
- Después se agrupan todos los pares con la misma clave de todas las listas, crea un grupo por cada una de las diferentes claves generadas.

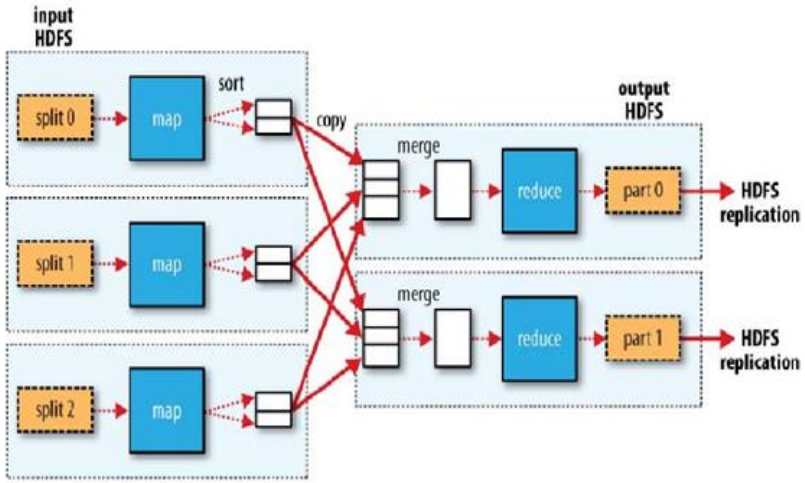
Map ($clave_1$, $valor_1$) –> lista ($clave_2$, $valor_2$)

- La operación de Map se paraleliza,

El conjunto de archivos de entrada se divide en varias tareas llamado FileSplit,

- Las tareas se distribuyen a los nodos TaskTrackers, y estos a su vez pueden realizar la misma tarea si hiciera falta.

Función Reduce( )



Funcionamiento de MapReduce

N archivos de entrada generará M mapas de tareas para ser ejecutados y cada mapa de tareas generará tantos archivos de salida como tareas Reduce hayan nconfiguradas en el sistema.

La función Reduce se aplica en paralelo para cada grupo creado por la función Map().

La función Reduce se llama una vez para cada clave única de la salida de la función Map.

Junto con esta clave, se pasa una lista de todos los valores asociados con la clave para que pueda realizar alguna fusión para producir un conjunto más pequeño de los valores.

Reduce (clave$_2$, lista(valor$_2$)) –> lista(valor$_2$)

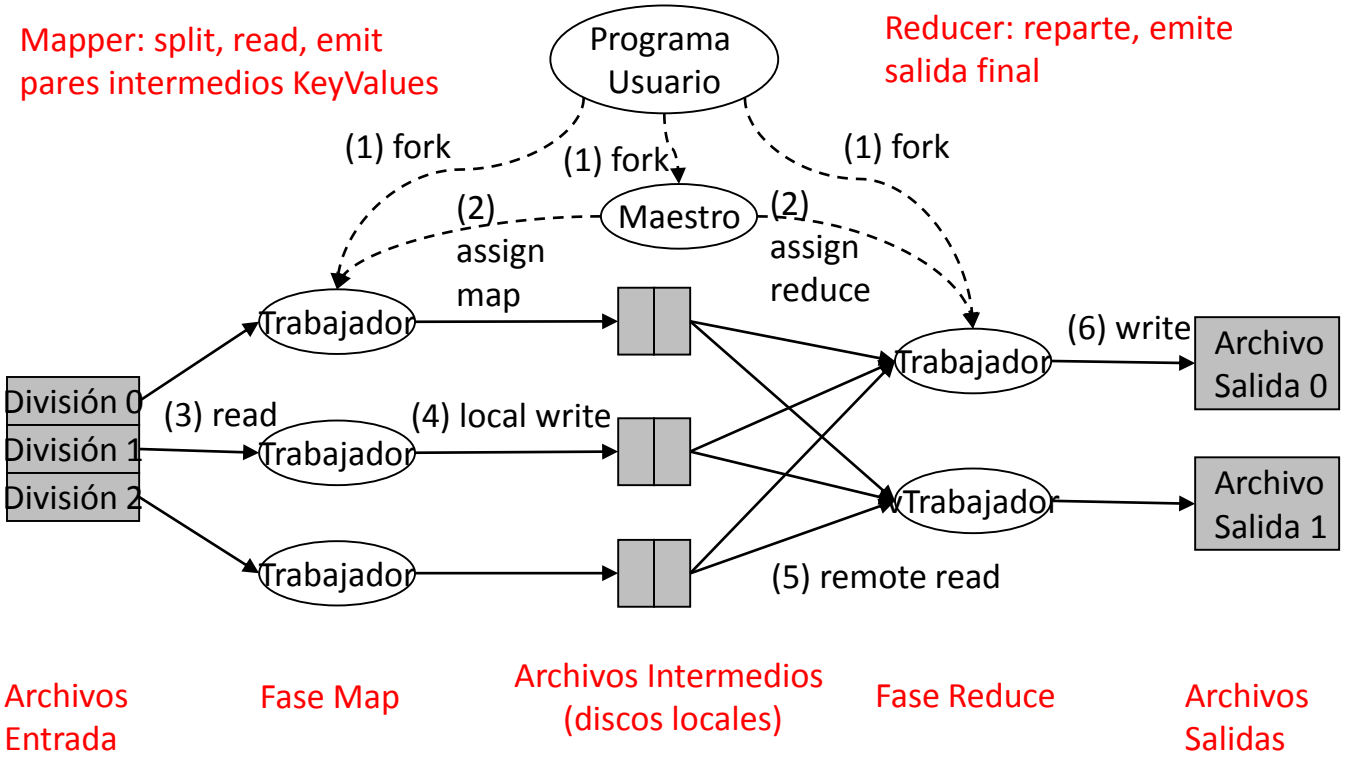
Cuando se inicia la tarea reduce, la entrada se encuentra dispersa en varios archivos a través de los nodos en las tareas de Map.

Los datos obtenidos de la fase Map se ordenan para que los pares clave-valor sean contiguos (fase de ordenación, *sort fase*).

Una vez que todos los datos están disponibles a nivel local, el archivo se fusiona (*merge*) de forma ordenado.

Al final, la salida consistirá en un archivo de salida por tarea reduce ejecutada.

Google MapReduce



Conteo de palabras:

Contar las apariciones de cada palabra en un conjunto de documentos:

Ejemplo

Map

```
map(String name, String document):
  // clave: nombre del documento
  // valor: contenido del documento
  for each word w in document:
    EmitIntermediate(w, 1);
```

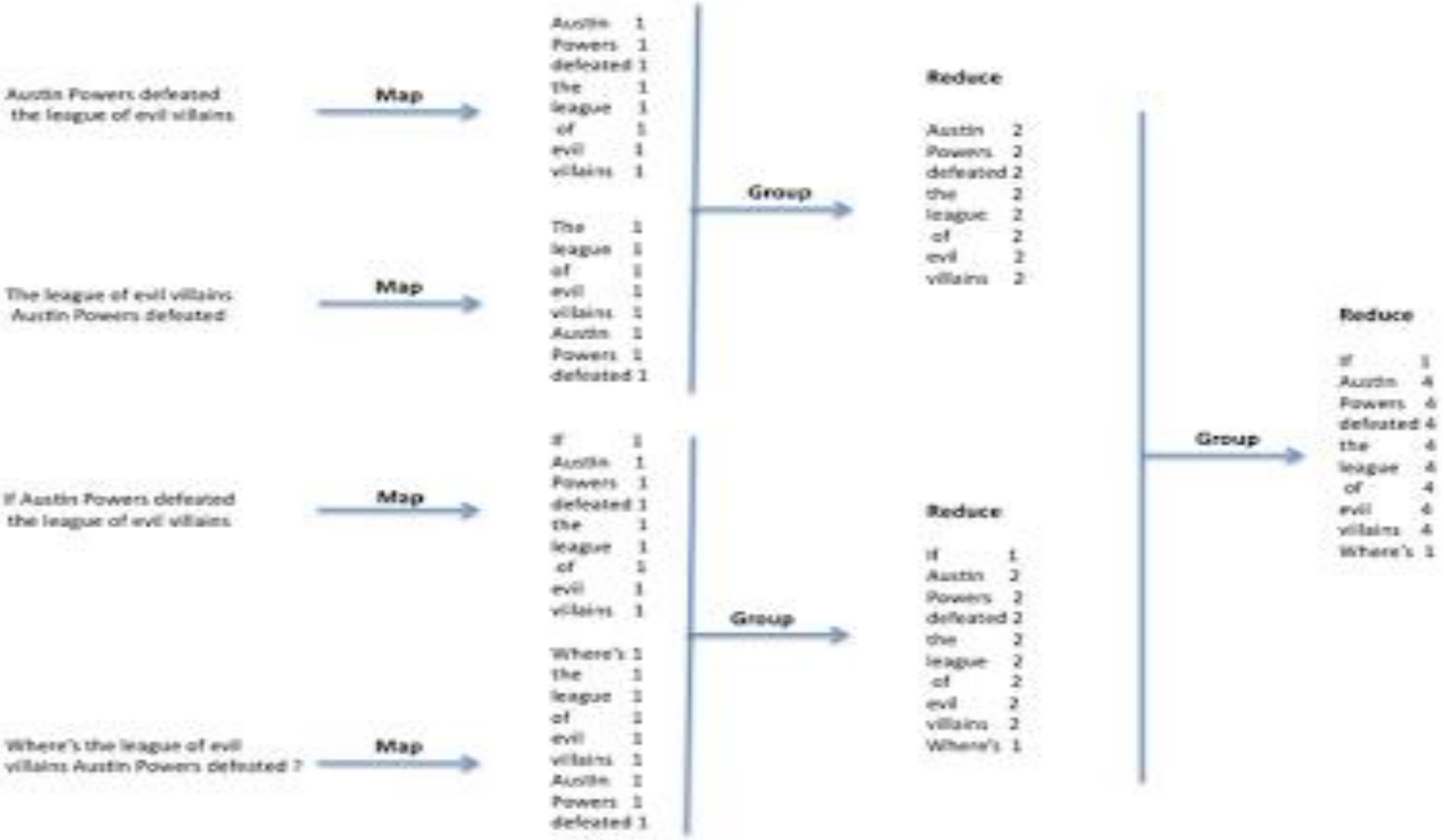
La función map() en este caso divide un documento en palabras (es decir lo *tokeniza*) mediante el empleo de un analizador léxico, y *emite* una serie de tuplas de la foma (*clave*, *valor*) donde la clave es la palabra y el valor es "1". Es decir, por ejemplo, del documento "La casa de la pradera" la función map retornaría: ("la", "1"), ("casa", "1"), ("de", "1"), ("la", "1"), ("pradera", "1").

Reduce

```
reduce(String word, Iterator partialCounts):
   // word: una palabra
  // partialCounts: una [[Iterador (patrón de diseño)|lista parcial]]
  //para realizar cuentas agregadas
  int result = 0;
  for each v in partialCounts:
        result += ParseInt(v);
  Emit(result);
```

El framework reúne todos los pares con la misma clave y alimenta a la misma llamada Reduce, para sumar todos los valores de su entrada para encontrar el total de las apariciones de esa palabra.

Contar la frecuencia de palabras a través de varios documentos



MapReduce

Otro ejemplo clásico del uso de MapReduce es el análisis de archivos logs.

- Grandes archivos se dividen y un **mapper** busca las diferentes páginas web que se accede.
- Cada vez que una página web se encuentra en el log, se emite al **reducer** un par clave/valor donde la clave es la página web y el valor es "1".
- Los **reducers** agregan el número de visualizaciones para ciertas páginas web.
- El resultado final es el número total de accesos para cada página web.

1. La función **map** procesa los logs de las solicitudes a página web y su salida es el par <URL, 1>.
2. La función **reduce** suma todos los valores para el mismo URL y emite un par <URL, total>.

MapReduce

GREP distribuido

- Es la tarea de encontrar un patrón en un número muy grande de documentos.
- Se puede utilizar, por ejemplo, para buscar en un conjunto de documentos, las líneas que coincidan con una expresión regular.

1. La función Map toma un par (archivo de entrada, línea) como entrada y genera una clave [(línea, 1)] si se encuentra una coincidencia.
2. La función de reduce toma como pares de entrada (línea, [1, 1, 1, ...]) y genera como salida (línea, n), donde "n" es el número de 1 de la lista.

MapReduce

Otras aplicaciones

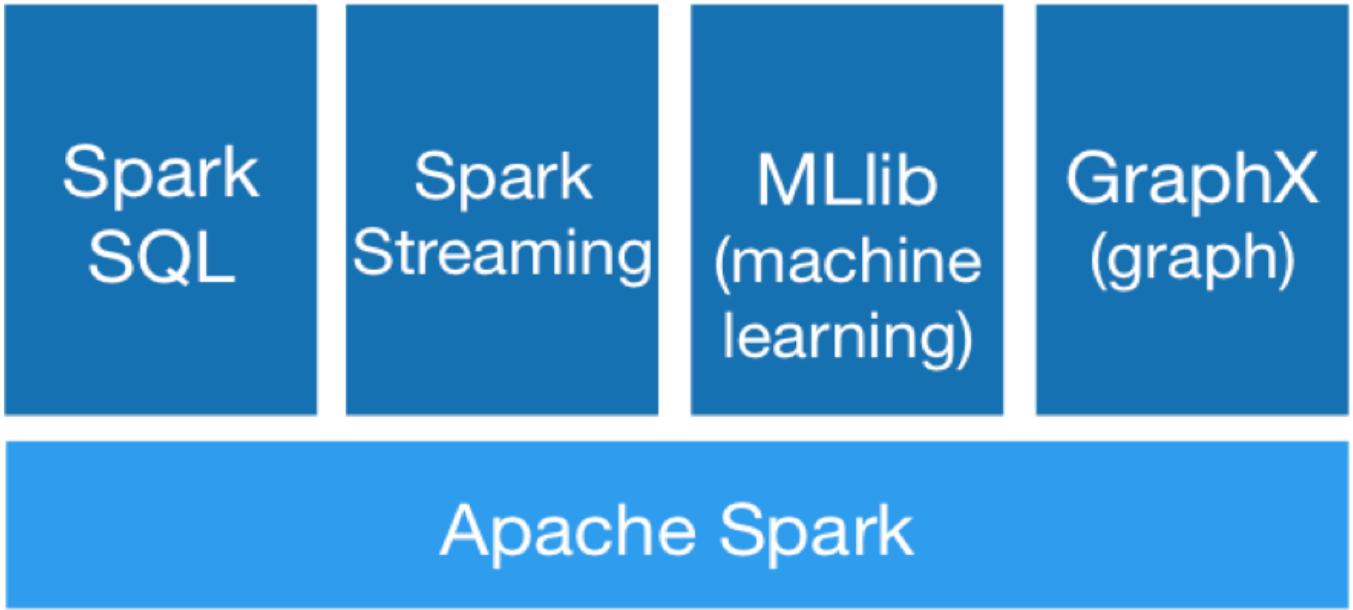
- Invertir en la web Web-Link Gráfo
- Encontrar posibles amigos en las redes sociales
- distribuir ordenamiento
- vector términos por host

Apache Spark

Es un potente motor de procesamiento de propósito general con gran velocidad, facilidad de uso y para análisis sofisticados.

- Spark tiene varias ventajas en comparación con otras tecnologías de big data.

  un marco integral y unificado para gestionar requisitos big data de procesamiento de una gran variedad de datos (texto, gráfos ..), así como de fuente (por lotes v. En tiempo real) .



Spark Core

Es el motor de ejecución y todas las demás funcionalidades y extensiones se construyen en la parte superior de la misma.

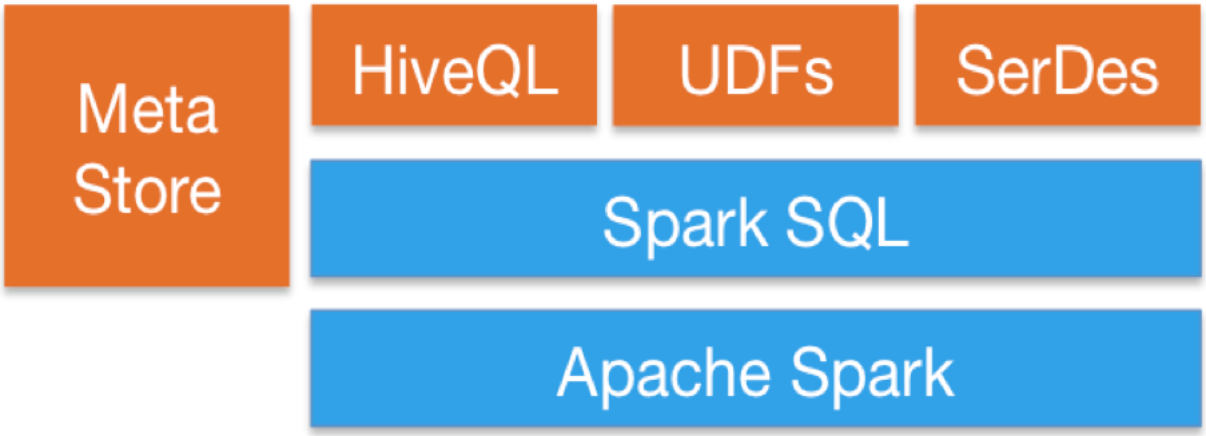
- ofrece en memoria capacidades de computación para ofrecer velocidad, un modelo de ejecución generalizada para soportar una amplia variedad de aplicaciones, y lAPI de Java, y Python para la facilidad de desarrollo.

- **Spark tiene varias características que lo diferencian de los motores tradicionales como MapReduce Hadoop:**
  - Es compatible con los DAG de cálculo generales más allá de la topología de MapReduce en dos etapas.
  - El motor de ejecución puede tolerar la pérdida de cualquier conjunto de nodos de trabajo y puede automáticamente volver a ejecutar tareas perdidas.
  - Proporciona una abstracción de almacenamiento en memoria de conjuntos de datos denominada *Data Distributed Resilient (DDR)* que permite a las aplicaciones guardar los datos en la memoria, y automáticamente reconstruye particiones perdidas en fracasos.
  - Se integra con un shell de línea de comandos Scala para permitir las operaciones interactivas sobre DDR.
  - DDR permite la especificación opcional de un partidor de datos, y el motor de ejecución puede explotar esa DDR.

Spark Core

Spark SQL es una biblioteca usada para consultar datos estructurados dentro de los programas Spark, utilizando SQL o API.

Spark SQL utiliza la interfaz Hive y MetaStore, que le da una compatibilidad total con los datos existentes en Hive, consultas y UDF.



Spark

Spark GraphX

- API para el cálculo de grafos en paralelo (por ejemplo, PageRank y el filtrado colaborativo).
- GraphX incluye una creciente colección de algoritmos de grafos para simplificar las tareas de análisis

Spark MLlib

- Algoritmos de aprendizaje automático y minería de datos, de alta calidad (por ejemplo, múltiples iteraciones para aumentar la precisión) y velocidad (hasta 100 veces más rápido que MapReduce).
- La biblioteca se puede utilizar en Java, Scala, y Python.
- Ejemplos de algoritmos:
  - Regresión logística
  - Máquinas de soporte vectorial (SVM)
  - k-medias, o la y reglas de asociación

Marcos de procesamiento de grafos

PEGASO

- Es una biblioteca de minería de grafos de código abierto
- Usa MapReduce con el fin de manejar operaciones de minería a gran escala.

Pregel

- es un modelo computacional adecuado para problemas de procesamiento de grafos a gran escala.

JaBeJa

- Es un algoritmo de partición completamente descentralizado basado en el algoritmo de Kernighan y Lin.
- Cada vértice en el grafo se asigna inicialmente a una partición aleatoria.
- En cada iteración, se trata de intercambiar los vecinos,
- Mueve las decisiones desde el nivel de partición a nivel de vértice, lo que aumenta en gran medida la posibilidad de la paralelización.

NoSQL

NoSQL (Not Only SQL): bases de datos que “van más allá de los" modelos de datos relacionales (es decir, no hay tablas, un uso limitado o nulo de SQL)

- Centrarse en la recuperación de datos y añadir nuevos datos (no necesariamente tablas)
- Centrarse en los almacenes de datos basado en claves que se pueden utilizar para localizar los objetos de datos
- Centrarse en apoyar el almacenamiento de grandes cantidades de datos no estructurados
- No ACID (atomicidad, coherencia, aislamiento, durabilidad)

NoSQL

NoSQL se centra en una arquitectura sin esquema (es decir, la estructura de datos no está predefinida)

- Las BDs relaciones tradicionales requieren el esquema definido antes de construir la base de datos.
  - Los datos se estructuran
  - Limitada en su alcance
  - Diseñado en torno a principios ACID.

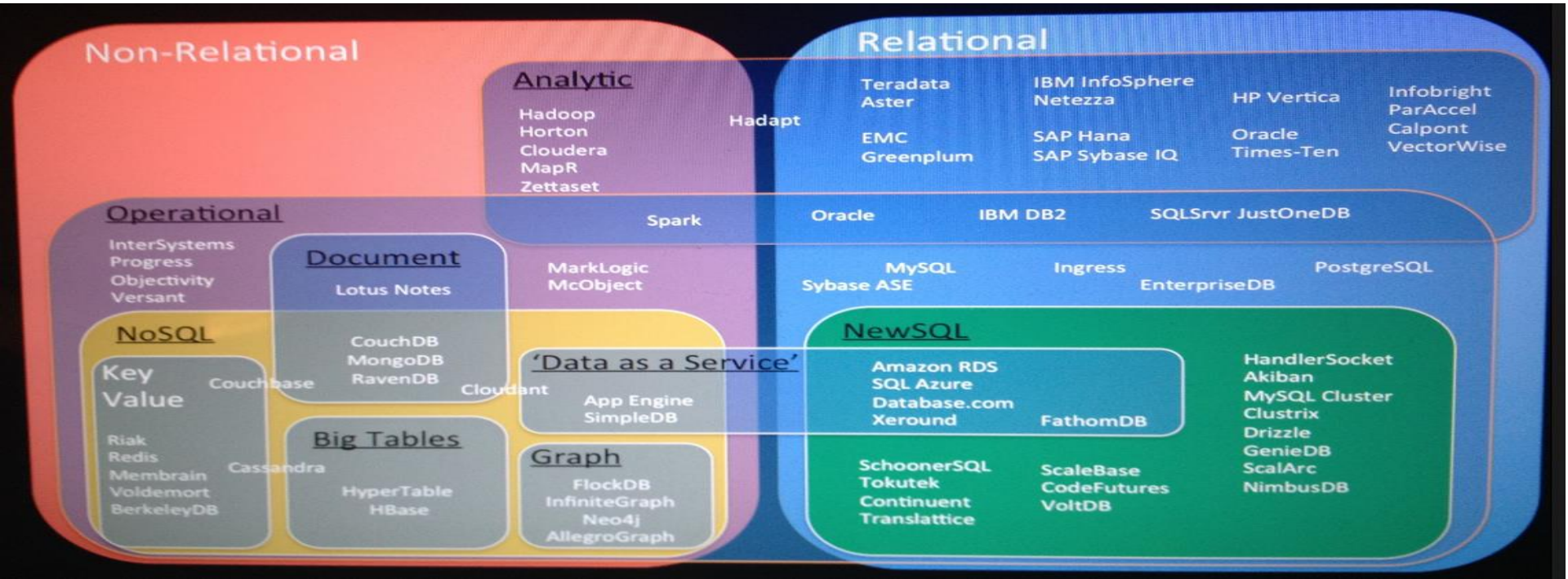
RDBMS vs. NoSQL

- Las bases de datos relacionales tradicionales nos permiten definir la estructura de un esquema que demanda reglas rígidas y garantizan ACID
- Las aplicaciones web modernas presentan desafíos muy distintos a las que presentan los sistemas empresariales tradicionales (e.j. sistemas bancarios):
  - Datos a escala web
  - Alta frecuencia de lecturas y escrituras
  - Cambios de esquema de datos frecuentes
  - Las aplicaciones sociales (no bancarias) no necesitan el mismo nivel de ACID

Algunas de las opciones de NoSQL actualmente disponibles son: Cassandra, MongoDB, Jackrabbit , BigTable y Dynamo

BASES DE DATOS RELACIONALES Y BASE DE DATOS NO RELACIONALES

Base de datos NO relacionales	Base de datos relacionales
Los datos almacenados no requieren estructuras fijas como tablas, no garantizan completamente **ACID** (**a**tomicidad, **c**oherencia, **a**islamiento y **d**urabilidad), y habitualmente escalan bien horizontalmente.	Permiten establecer interconexiones (relaciones) entre los datos (que están guardados en tablas), y a través de dichas conexiones relacionar los datos de ambas tablas



¿Por qué necesitamos NoSQL?

- **Las BBDD relacionales ofrecen bajo rendimiento ante ciertas aplicaciones intensivas de datos:**
  - Indexación de un gran número de documentos
  - Servir páginas en sitios de mucho tráfico
  - Envío de datos de streaming

- **Las RDBMS están optimizadas para pequeñas pero frecuentes transacciones de lectura/escritura, o largas transacciones con pocos acceso de escritura.**

- **NoSQL puede dar servicio a grandes cargas de lectura/escritura:**
  - Digg mantiene 3 TB de green badges (marcadores que indican las historias votadas por otros en una red social)
  - Facebook tiene que realizar búsqueda en bandejas de mensajes de más de 50 TB

Taxonomía de soluciones NoSQL

- NoSQL pueden clasificarse en función de su modelo de datos en las siguientes cuatro categorías:

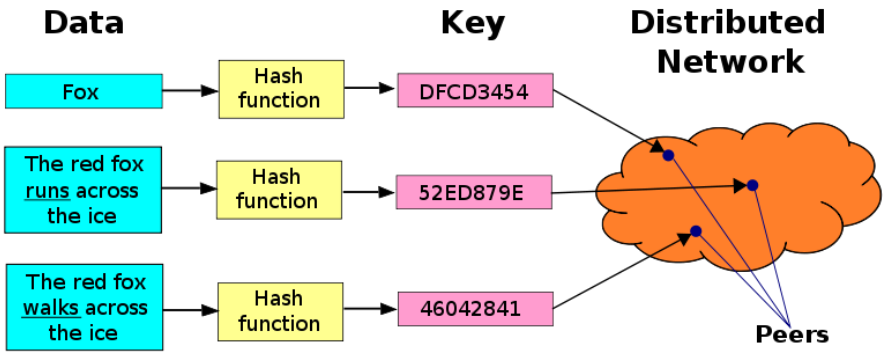
1. **Orientadas a clave-valor (Key-Value stores)**
2. **Orientadas a columnas (Wide Column stores)**
3. **Orientadas a documentos (Document stores)**
4. **Orientadas a grafos (Graph databases)**

1. BBDD orientadas a Clave-Valor

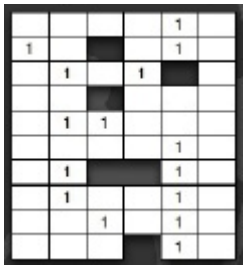
- Su precursor fue Amazon Dynamo
  - Basadas en DHT (Distributed Hash Tables)
- Modelo de datos: colección de pares: clave/valor
- Ejemplos: Dynomite, Voldemort, Tokyo

Un **DHT** es una clase de sistema distribuido que permite un servicio de lookup similar a un Hash Table

- Almacenan pares clave valor
- Cada nodo puede obtener el valor asociado a una clave
- La responsabilidad de mantener los mapeos entre claves y valores está distribuida entre los nodos
- Escalan a grandes números de nodos

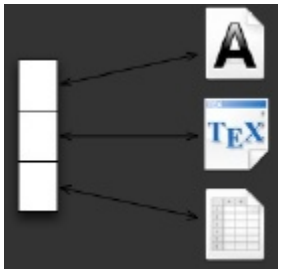


2. BBDD orientadas a Familia de Columnas

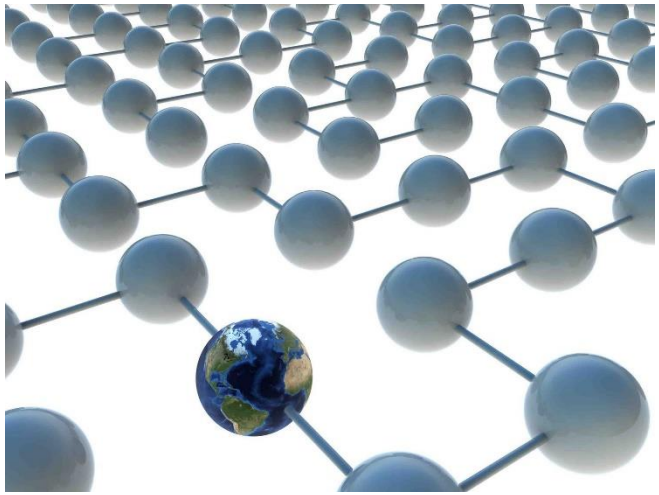


- Su precursor es Google BigTable
- **Modelo de datos:** familia de columnas, esto es, un modelo tabular donde cada fila puede tener una configuración diferente de columnas
- Ejemplos: HBase, Hypertable, Cassandra, Riak
- Buenas en:
  - Gestión de tamaño
  - Cargas de escrituras masivas orientas al stream
  - Alta disponibilidad
  - MapReduce

3. BBDD orientadas a Documentos



- La precursora fue Lotus Notes
- Modelo de datos: colecciones de documentos que contienen colecciones de claves-valor
- **Ejemplos:** CouchDB, **MongoDB**
- Buenas en:
  - Modelado de datos natural
  - Amigables al programador
  - Desarrollo rápido
  - Orientas a la web: CRUD

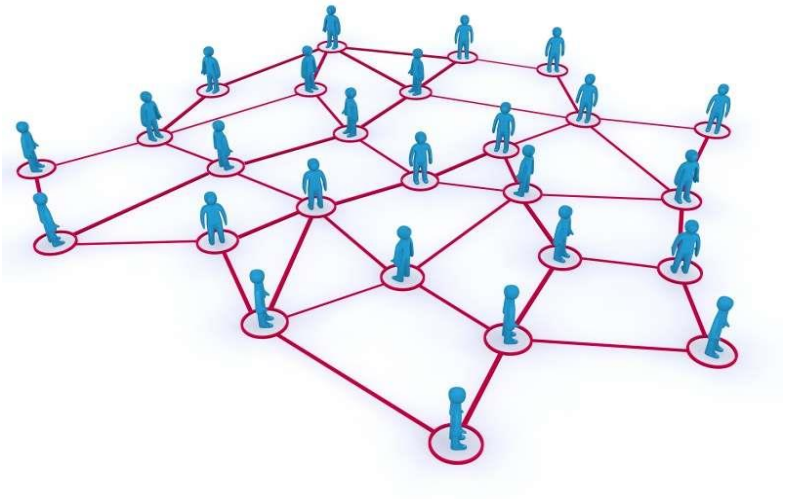


Un documento encapsula información en un formato estándar (XML, YAML, JSON)

- Los documentos en una BBDD orientada a documentos son similares a registros pero no requieren un esquema estándar con la mismas secciones y partes
- Los documentos suelen ser direccionables por una clave que los representa unívocamente
- Además de la búsqueda por clave de documento, estas BBDD suelen ofrecer una API o lenguaje de consultas que permite recuperar documentos en base a sus contenidos

4. Bases de Datos Basadas en Grafos

- Inspiradas por la teoría de grafos
- Modelo de datos: nodos, relaciones entre pares de nodos, valor clave en ambos
- Ejemplos: AllegroGraph, VertexBD, Neo4j



Análisis de Big Data

Es el proceso de examinar grandes cantidades de datos de una variedad de tipos para descubrir patrones ocultos, correlaciones desconocidas y otra información útil.

Análisis de Big Data

Tipos

Tipo de Análisis	Descripción
Analítica básica para la comprensión	Reordenamiento de datos, presentación de informes, monitoreo básico.
Analítica avanzada para la visualización	Análisis complejo, como el modelado predictivo y otras técnicas de comparación de patrones.
Análisis operacional	Analítica se convierte en parte del proceso de negocio.
Análisis monetizado	Analítica se utilizan para generar ingresos.

Ecosistema de Hadoop

Apache Pig

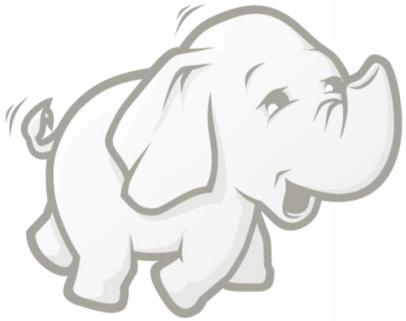
High-level language for expressing data analysis programs

Apache Hive

SQL-like language and metadata repository

Apache HBase

The Hadoop database. Random, real -time read/write access



Hue

Browser-based desktop interface for interacting with Hadoop

Apache Zookeeper

Highly reliable distributed coordination service

Oozie

Server-based workflow engine for Hadoop activities

Flume

Distributed service for collecting and aggregating log and event data

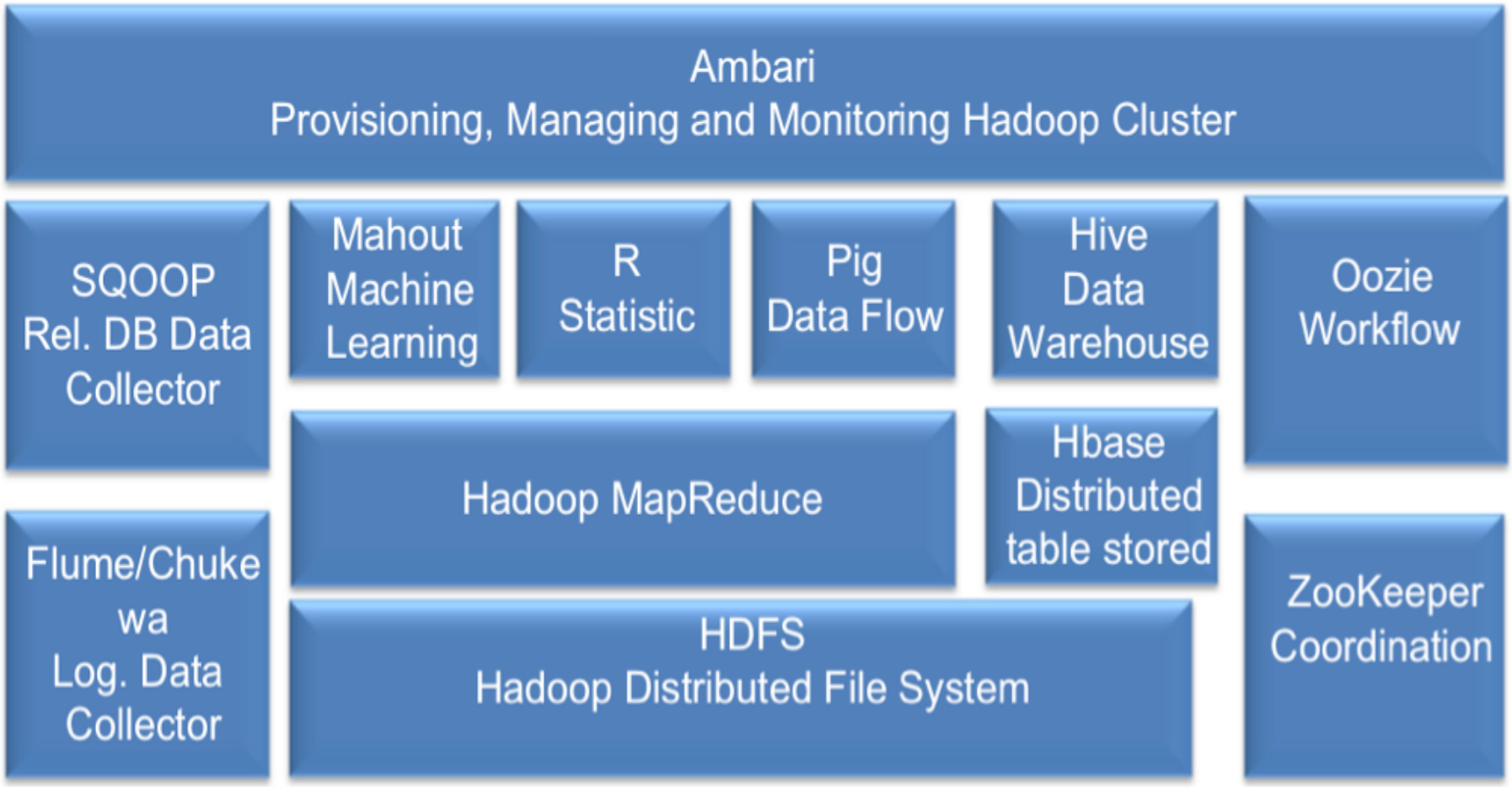
Sqoop

Integrating Hadoop with RDBMS

Apache Whirr

Library for running Hadoop in the cloud

Ecosistema Hadoop



Ecosistema Hadoop

Ofrecen soporte y funcionalidades a los desarrolladores de aplicaciones Hadoop

AVRO: es un sistema para la serialización de datos y uno de los principales métodos para transportar información entre las herramientas y aplicaciones Hadoop.

ZOOKEEPER: es un servicio centralizado que se encarga de administrar y gestionar la coordinación entre procesos en sistemas distribuidos.

SOLR: Apache Solr es un motor de búsqueda basado en el Apache Lucene, escrito en Java y que facilita a los programadores el desarrollo de aplicaciones de búsqueda.

Ecosistema Hadoop

Recolección de datos

CHUKWA: es una herramienta principalmente pensada para trabajar sobre logs y realizar análisis. Está construido por encima de HDFS y MapReduce, por lo que hereda su escalabilidad y robustez.

FLUME: es una herramienta distribuida para la recolección, agregación y transmisión de grandes volúmenes de datos. Ofrece una arquitectura basada en la transmisión de datos por streaming altamente flexible y configurable, pero a la vez simple.

Ecosistema Hadoop

Almacenamiento

Aunque Hadoop ya cuenta con su propio sistema de almacenamiento, HDFS, existen varios proyectos que complementan, e incluso trabajan conjuntamente, con el sistema de ficheros para aumentar las características y la flexibilidad de Hadoop entorno a las necesidades de sus usuarios.

CASSANDRA: es una base de datos NoSQL, mayormente desarrollada por Datastax aunque empezó como un proyecto de Facebook, escrita en Java y open-source, también es un proyecto Apache.

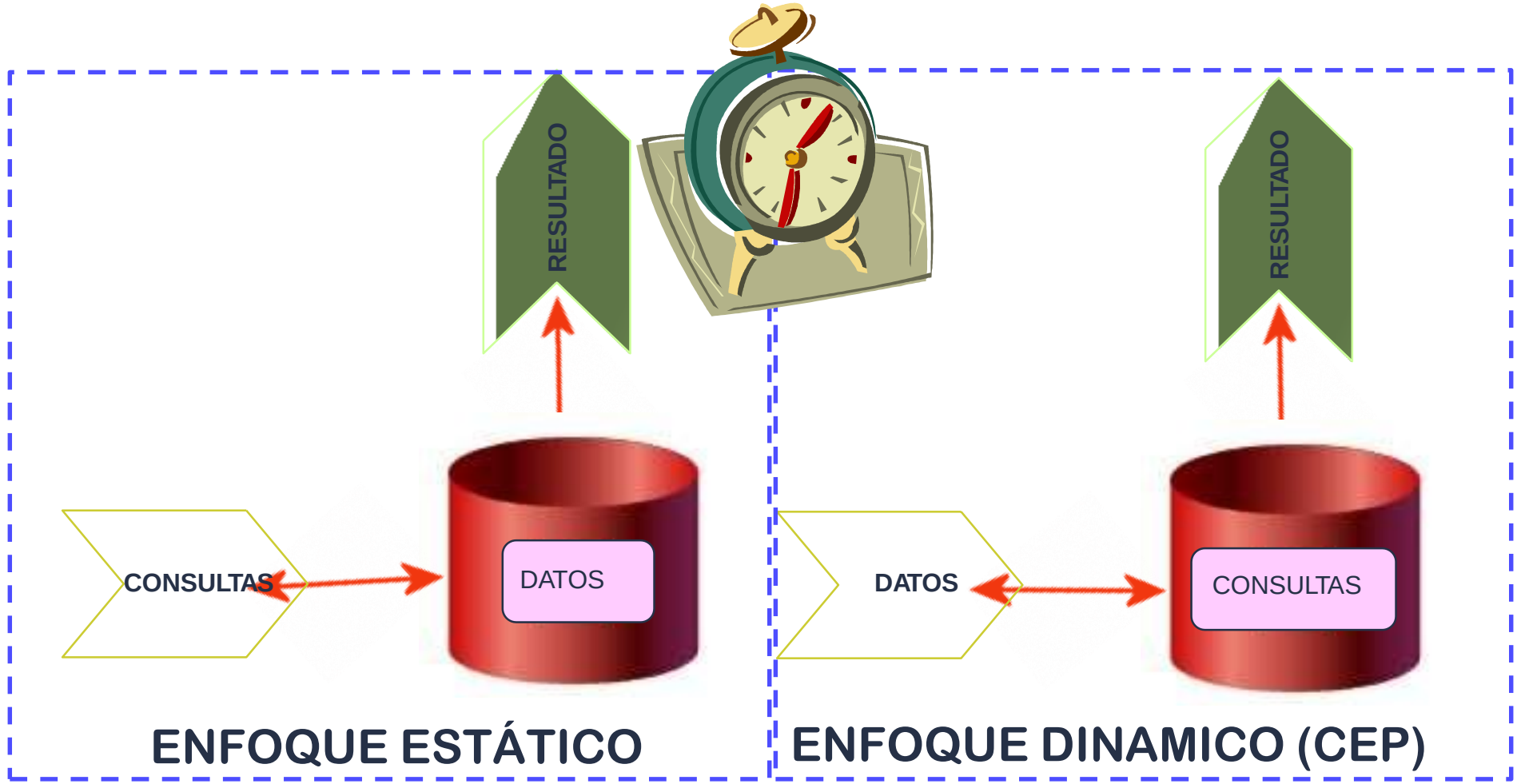
HIVE: es una herramienta para data warehousing que facilita la creación, consulta y administración de grandes volúmenes de datos distribuidos en forma de tablas relacionales.

TIEMPO REAL EN BIG DATA: Fast Data

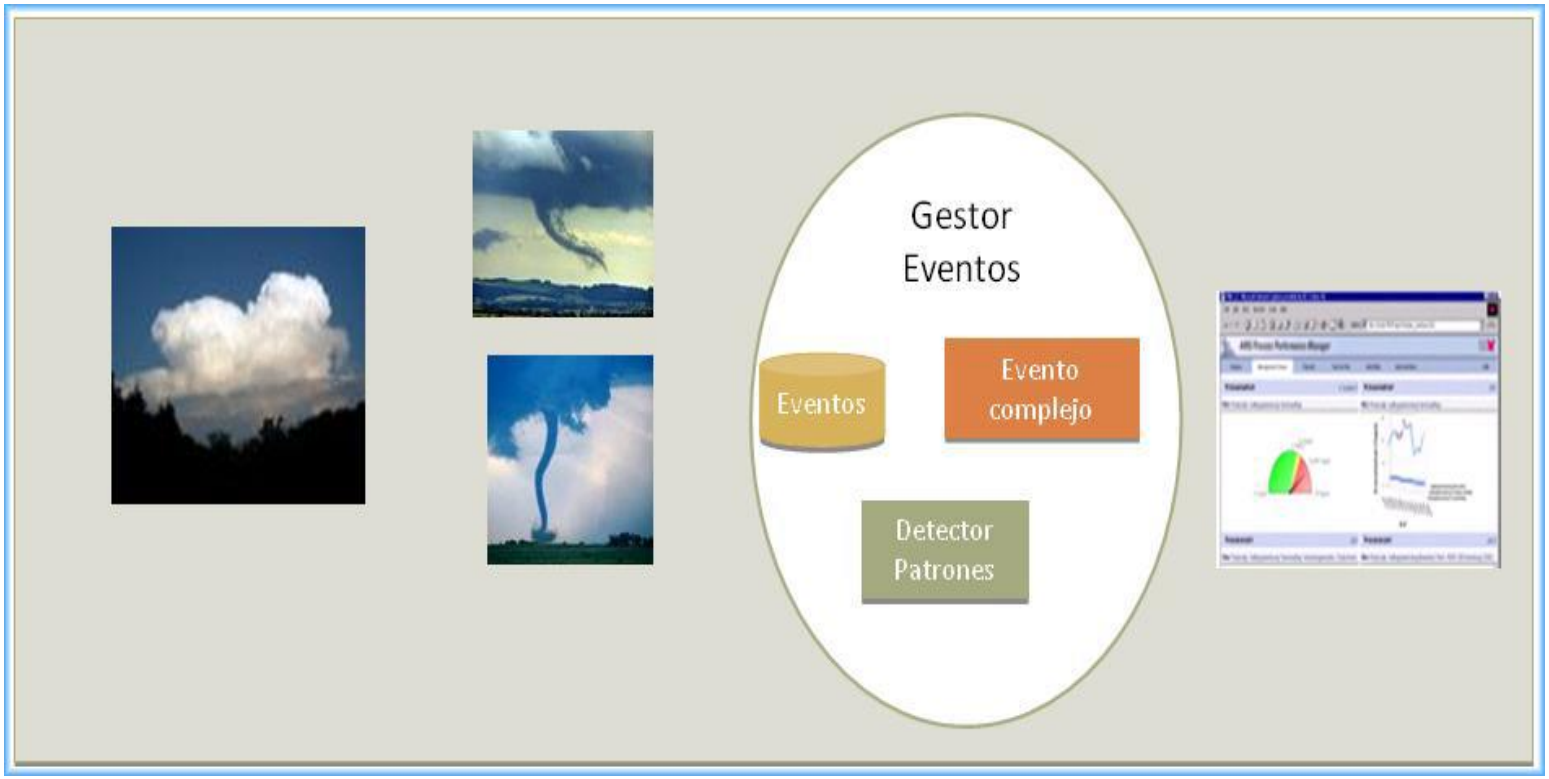
- Cuando procesas gran cantidad de datos interactuando en un determinado periodo de tiempo, la gestión de los mismos y la capacidad de **“separar el grano de la paja”,** se vuelve cada vez más difícil y complejo.

- **Fast Data** nos permite utilizar tecnologías dinámicas como **CEP (Procesamiento de Eventos Complejos**) para, en tiempo real, actuar sobre los datos y obtener **patrones**

Procesamiento estático vs dinámico (CEP)



Gestor de Eventos Complejos



TIEMPO REAL EN BIG DATA: Fast Data

Gestor de Eventos Complejos

Técnicas

Data Stream - ESP (Event Stream Processing):

La velocidad es su máxima preocupación y lo que permite es realizar un análisis mientras los datos están en movimiento, antes de que lleguen a su destino final, momento en el que ya se guarda toda la información obtenida.

Un ejemplo de aplicación serían los sistemas de gestión de flujos de datos (Data Stream Management System, DSMS) en el ámbito de las redes de sensores.

Un DSMS es un sistema que ejecuta consultas continuas sobre un flujo continuo de datos (data stream). Estos datos, que llegan en cada momento, permanecen sólo por un corto periodo de tiempo en memoria.

Procesamiento Dinámico CEP (Complex Event Processing) :

Se trata de un análisis en tiempo real, pero centrándose en los eventos que puedan suceder en un determinado proceso, es decir, su objetivo es buscar eventos pre-definidos, analizando los datos en tiempo real.

Ejemplos

Analiza en tiempo real todo lo que está sucediendo con mis tarjetas de crédito en cajeros y comercios.

Analizar en tiempo real la información que generan sensores, para alertar antes de que falle una máquina.

EPL: Event Pattern Language

Además de un Gestor de Eventos Complejos, se necesita un lenguaje de programación, que permita no solo la consulta y el procesamiento de eventos, sino la posibilidad de añadir una "dimensión temporal", y lo que es más importante inferir nuevos eventos basados en patrones semánticos.

- Este lenguaje se conoce como EPL (Event Pattern Language).
- EPL amplía y extiende a SQL en la definición de reglas temporales y en la aplicación de reglas no lineales para el procesamiento de eventos.

Permiten:

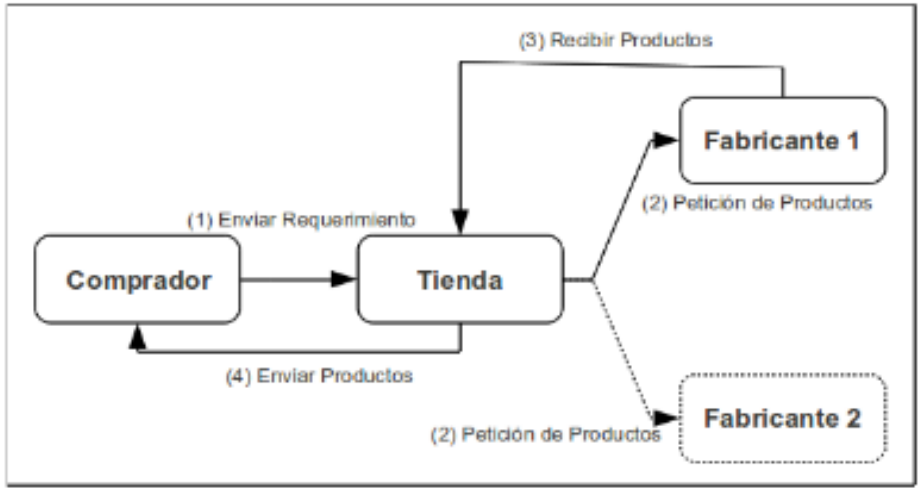
- Filtrar eventos en función de una ventana temporal,
- Crear nuevas corrientes de eventos a partir de una o varias corrientes existentes, y
- Definir patrones que faciliten aplicar reglas de negocio a los

Algunos ejemplos de EPL"s son:

- Coral8 CCL
- EQL (Event Query Languaje), similar a SQL, eventos capturados.
- RAPIDE-EPL
- StreamSQL

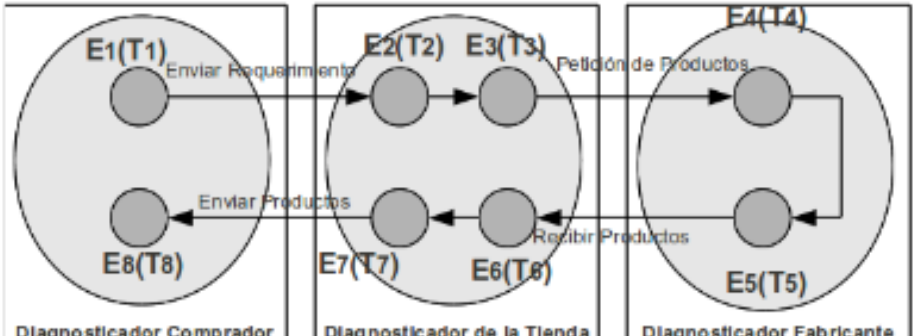
Caso de estudio

- Industria del Mueble

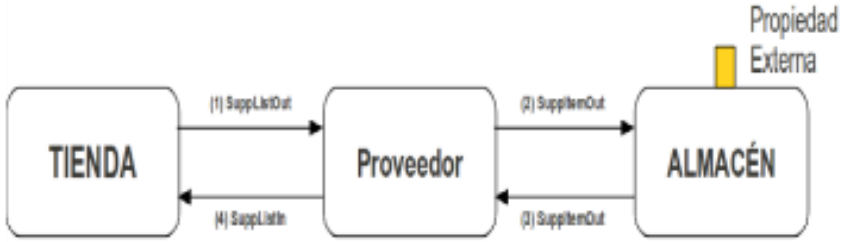


tuneQualityBehavior: generar la falla en E5

turnprovider: Fabricante a invocar (Fabricante 1 o Fabricante 2)



- Comercio Electrónico

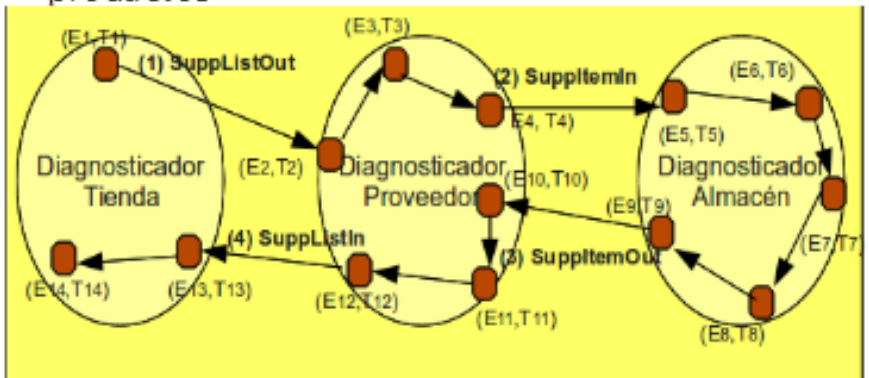


tuneSearchBehavior: "Buscar Productos" no produce resultados).

tuneShip: Falla en E8 (productos sin empaquetar).

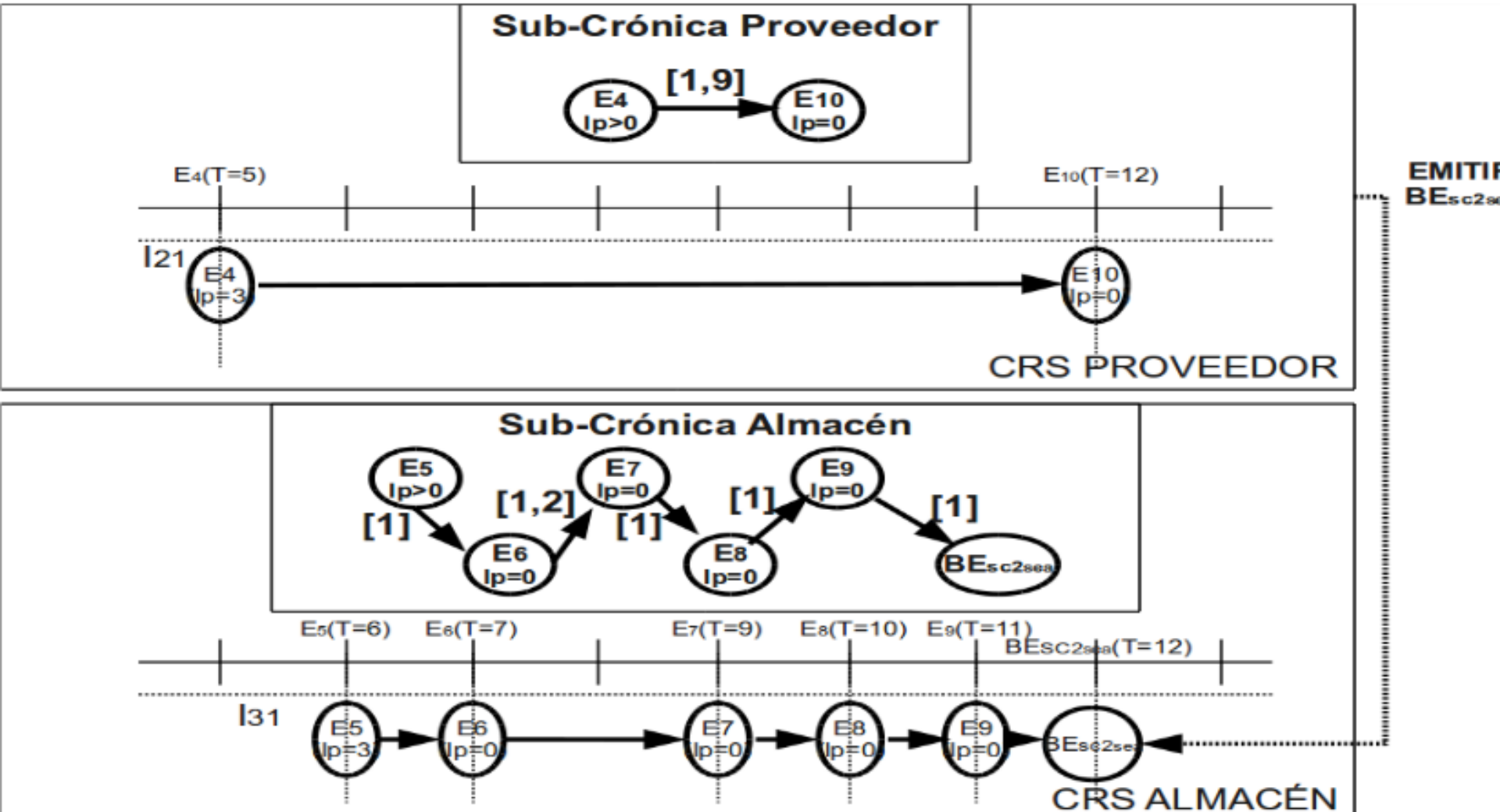
setTuneDelay: Induce retardos

ExternalSearch: Permite busqyeda externa de productos



Caso de estudio

Pruebas de Mecanismo de reconocimiento Distribuido (Violación de SLA por parte del Almacén)



Caso de estudio

Chonicle Model SLA Violation Warehouse

Subchronicle Shop {
Events{
}
Constrains{
}
When recognized{
}
}

Subchronicle Supplier {
Events{
event(e_{23}: pl = 3, T_{23}),
event(e_{24}: pl = 0, T_{24})
}
Constrains{
T_{24}-$T_{23} \leq 10$
}
When recognized{
Emit event(E_{SC2Sea}, T_{SC2Sea}, D3)
}
}

Subchronicle Warehouse {
Events{
event(e_{31}: pl = 3, T_{31}),
event(e_{32}: pl = 0, T_{32}),
event(e_{33}: pl = 0, T_{33}),
event(e_{34}: pl = 0, T_{34}),
event(e_{35}: pl = 0, T_{35}),
event(e_{SC2Sea}, T_{SC2Sea})
}
Constrains{
T_{32}-$T_{31} \leq 2$
T_{33}-$T_{32} \leq 2$
T_{34}-$T_{33} \leq 3$
T_{35}-$T_{34} \leq 1$
T_{SC2Sea}-$T_{35} \leq 3$
}
When recognized{
repairer Invoke(Warehouse, 'SLA-Violation')
}
}

Event Pattern Language

(Construcción de crónicas usando el lenguaje CQL)

Falla de violación de SLA en el Almacén (caso Búsqueda de Productos)

```
Subchronicle Proveedor SLA{
  SELECT ISTREAM(id => E4.id, event => 'BESLA', time
=> E10.time, Ipsupplier => E10.Ip, Ip => E4.Ip, to =>
'Diagnoser Almacén')
FROM
   E4[10], E10[now]
WHERE
   E10.time >= E4.time AND E10.id = E4.id AND E4.Ip -
E10.Ip > 0
}
```

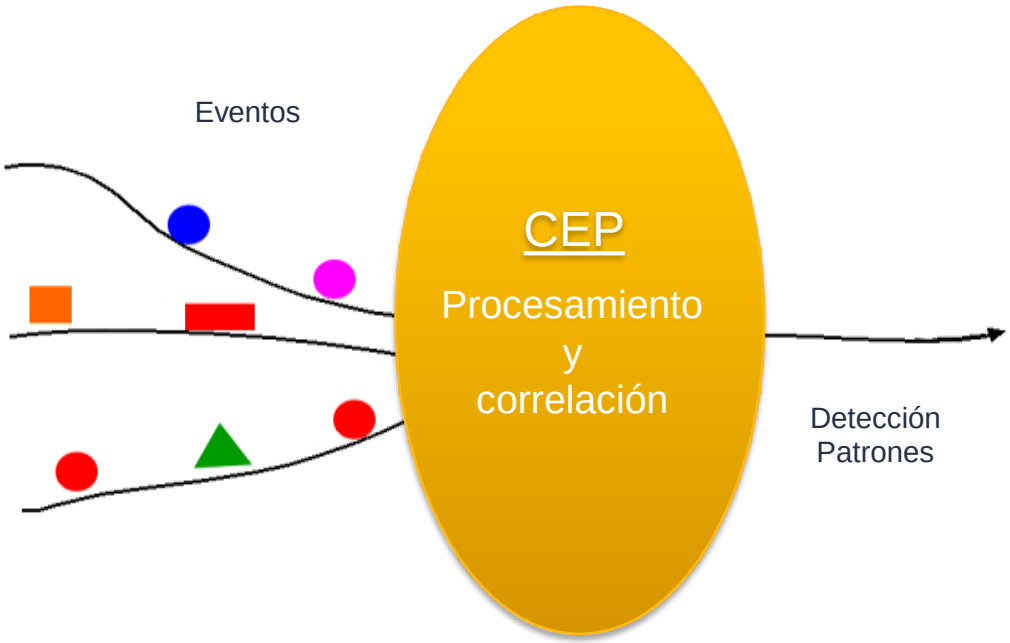
```
Subchronicle Almacén SLA Búsqueda de Productos {
  SELECT  ISTREAM( id => E5.id,  fault =>
'ServiceLevelAgreement', faulttype => 'SearchProducts',
     time => EBESLA.time,  Ip => E6.Ip, to => 'Repair
Almacén')
FROM
   E5[10], E6[8], E7[7], E8[5] E9[3], EBESLA[now]
WHERE
   E6.time >= E5.time AND  E7.time >= E6.time AND
   E8.time >= E7.time AND  E9.time >= E8.time AND
   EBESLA.time >= E9.time AND  E5.Ip > 0 AND
   E6.Ip = 0 AND  E6.Ip = E7.Ip AND E8.Ip = E7.Ip AND
   E6.id = E5.id AND   E7.id = E6.id AND E8.id = E7.id
AND   EBESLA.id = E7.id
}
```

Falla de violación de SLA en el Almacén (caso de Empaqueta y envía)

```
Subchronicle Proveedor SLA{
  SELECT ISTREAM(id => E4.id, event => 'BESLA', time
=> E10.time, Ipsupplier => E10.Ip, Ip => E4.Ip, to =>
'Diagnoser Almacén')
FROM
   E4[10], E10[now]
WHERE
   E10.time >= E4.time AND E10.id = E4.id AND E4.Ip -
E10.Ip > 0
}
```

```
Subchronicle Almacén SLA Empaqueta y envía {
  SELECT ISTREAM(id => E5.id, fault =>
'ServiceLevelAgreement', faulttype => 'Ship and Packed',
time => EBESLA.time, Ip => E8.Ip, to => 'Repair 3')
FROM
   E5[10], E6[8], E7[7], E8[5], EBESLA[now]
WHERE
   E6.time > E5.time AND E7.time > E6.time AND E8.time
> E7.time AND EBESLA.time > E8.time AND E5.Ip > 0
AND E5.Ip = E6.Ip AND E6.Ip = E7.Ip AND E8.Ip < E7.Ip
AND E6.id = E5.id AND E7.id = E6.id AND E8.id = E7.id
AND EBESLA.id = E7.id
}
```

Aplicación de CEP



Trading

- Identificar automáticamente órdenes de compra y venta

Fraude

- Detectar compras fraudulentas en un corto intervalo de tiempo

CRM

- Correlacionar llamadas de atención al cliente con problemas de operación

Seguridad

- Procesar en tiempo real sensores y generar alarmas

ARQUITECTURA ORIENTADA A SERVICIOS (SOA) CON GESTOR DE EVENTOS COMPLEJOS